HSR Hochschule für Technik Rapperswil
Studienarbeit, FS 2009

Arbeitsthema:

Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur

retrofittable interface for integrating diagnosis devices into medical it infrastructure

Themengebiet:

Medizininformatik

Autor	Florian Vetter fvetter@hsr.ch
Betreuer	Prof. Dr. Axel Doering
Organisation	HSR Hochschule für Technik Rapperswil
Abteilung	Informatik
Institut	Institut für Software IFS
Studienjahr	Frühlingssemester 2009 (16.02.2009 – 13.09.2009)
Projektdauer	16.02.2009 – 29.05.2009
Abgabetermin	Fr 29.05.2009, 17:00Uhr

Einstiegspunkt Dokumentation

19. Mai 2009

Abstract

Dieses Dokument dient als Einstiegspunkt zur Dokumentation dieser Studienarbeit. Hier finden Sie die wichtigsten Informationen zur Studienarbeit inkl. Verweise auf Dokumente, welche inhaltlich nach ihren thematischen Schwerpunkte gegliedert sind.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Einstiegspunkt_Dokumentation_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
06.05.2009	Dokument erstellt
19.05.2009	Dokument in Version 2 überarbeitet
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument dient als Einstiegspunkt zur Dokumentation dieser Studienarbeit.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

In diesem Dokument werden die wichtigsten Informationen und Verweise auf Dokumente, welche inhaltlich nach ihren thematischen Schwerpunkte gegliedert sind. Zusammengefasst.

Es existiert weiter ein Management Summary, eine kurze Stellungnahme zum Stand der Projektarbeiten und ein Persönlicher Berichte des Studenten.

2 Management Summary

2.1 Ausgangslage

Mit DICOM (Digital Imaging and Communication in Medicine) und HL7 (Health Level 7) existieren wichtige Standards zum Austausch zwischen (meist bildgebenden) Medizin- und Laborgeräten und zentralen Informationssystemen (z.B. einem Klinik-Informationssystem KIS oder einem Labor-Informationssystem LIS).

Gegeben durch den stetig technologischen Wandel innerhalb der Medizinalbranche, stehen heutzutage vielerorts noch Medizin- oder Laborgeräte im Einsatz, welche nicht über die obigen erwähnten Standards verfügen, oder diese nur teilweise implementieren. Da diese Geräte oft ihre Produktlebensdauer noch nicht überschritten haben, werden Lösungen gesucht, die eine einfache Integration zu modernen Informationssystemen über Schnittstellen sicherstellen.

Ein beispielhaftes Anwendungsszenario ist der Import von Bild- und Patientendaten von einem bildgebenden Gerät unter Verwendung einer vorhandenen Druckfunktion. Mittels einer auf dem Gerät nachrüstbaren Komponente sollen diese Daten auf einem DICOM-Server archiviert werden.

Da viele bildgebende Diagnosegeräte Rechner mit verbreiteten Betriebssystemen beinhalten, oder an solche angeschlossen werden, ist damit eine sehr universelle Möglichkeit der Integration gegeben.

2.2 Ergebnisse

Die in dieser Studienarbeit erarbeitete Lösung sieht den Einsatz eines Client- Serversystems vor, welches die nötigen Schnittstellen clientseitig zu Verfügung stellt. Nach Angabe der Metainformationen (Patienten- und Studieninformationen) zur Beschreibung der Ressource (Bilddatei), leitet der Client diese für die Weiterverarbeitung an einen Broker weiter. Der Broker als zentrale Instanz nimmt diese Daten des Clients entgegen, verarbeitet und kapselt die Informationen als DICOM-Datei, welche er anschliessend an ein DICOM-Archiv sendet.

Für die Gewährleistung der DICOM-Konformität wird auf die freie Open Source Lösung dcm4che aufgesetzt, welche den DICOM-Standard in Java implementiert.

2.3 Ausblick

Es wurde ein Prototyp realisiert, der Bilddateien (z.B. *.jpg) an ein DICOM-Archiv exportiert. In einer weiteren Arbeit kann die Erstellung dieser Bilddateien über einen Druckertreiber ergänzt werden, so dass jegliche Druckausgaben einer bestimmten Applikation alternativ an ein DICOM-Archiv umgeleitet werden können.

3 Direkteinstieg in Dokumentation

Durch die Gliederung der Dokumentation nach inhaltlichen Schwerpunkten, sind entsprechende Teildokumente erstellt worden. Im nachfolgenden Dokumentverweis finden Sie eine Aufstellung all dieser Dokumente.

3.1 Dokumentverweis

Dokumentname	Inhaltsübersicht
Einstiegspunkt Dokumentation	Das Dokument, welches Sie gerade lesen.
DICOM Basics	Dieses Dokument gibt einen Einblick in die Welt des DICOM Standards.
Anforderungsanalyse	Dieses Dokument beinhaltet die Anforderungsanalyse mit Ist-/Soll-Zustand, funktionale und nicht funktionale Anforderungen, Szenarios, Use Cases, Domain Model und Aktivitätsdiagramm.
Anforderungsspezifikation	Dieses Dokument beinhaltet die Anforderungsspezifikation und Detailspezifikationen zu dieser Studienarbeit.
Architektur-Design	Dieses Dokument beschreibt das Architekturdesign mit den verschiedenen Überlegungen und Architekturentscheide bezüglich Implementierung dieser Softwarelösung.
Betreiberdokumentation	Dieses Dokument richtet sich an den Betreiber und Entwickler dieser Softwarelösung. Inhaltliche Schwerpunkte sind Installation und Konfiguration der Arbeit- und Entwicklungsumgebung.
Benutzerdokumentation	Dieses Dokument richtet sich an den Benutzer dieser Softwarelösung. Inhalt dabei ist die Installations- sowie Bedienungsanleitung.
Configurations Management Plan	Dieses Dokument beschreibt die Installation aller Softwarekomponenten, Konfiguration der Arbeitsumgebung, sowie alle zur Wiederherstellung eines bestimmten Projektstandes notwendigen Informationen.
Systemtests	Hier finden Sie alle Informationen zu den durchgeführten Systemtests, welche sich auf die Funktionalitäten aus den Anforderungsspezifikation beziehen.
Projektplanung	Hier finden Sie alle Informationen zur Planung dieser Studienarbeit.
Glossar	Im Glossar finden Sie alle verwendeten Fachwörter oder Abkürzungen mit jeweiliger Erklärung.
Anhang	(nachfolgende Dokumente befinden sich im Anhang)
Projektplan	Hier finden Sie den Projektplan zu dieser Studienarbeit.
Projektmonitoring	In diesem Dokument finden Sie eine Auswertung der aufgewendeten Arbeitsstunden für diese Studienarbeit sowie ein Review zum Projektverlauf.
Protokolle	Hier finden Sie die Protokolle zu den wöchentlich durchgeführten Sitzungen.

Table 1: Dokumentverweis

4 Stand der Projektarbeiten

4.1 Ist-Zustand

In dieser Studienarbeit wurde eine fertige Infrastruktur für die Projektarbeit aufgebaut. Diese beinhaltet die Entwicklungsumgebung auf der Arbeitsstation des Entwicklers, auf welchem ein DICOM-, sowie ein Application-Server als virtuelle Server betrieben werden.

Zusätzlich wurde für die Bearbeitung ein Trac mit integriertem SVN-Repository eingerichtet und ein Build-Server installiert.

Für die zu entwickelnde Lösung wurde ein Softwareprojekt für die Client- und Broker-Komponente eingerichtet und die entsprechenden Funktionalitäten in einem Prototypen implementiert und soweit wie möglich ausgebaut.

Der derzeitige Entwicklungszustand umfasst einen Client, welche über ein GUI das Erfassen und Bearbeiten der Aufträge für den DICOM-Import, sowie das Erfassen und Bearbeiten der entsprechenden Ressourcen mit Hinzufügen von Metainformationen ermöglicht.

Die DICOM-Sendefunktionalität ist in einer separaten Java-Klasse vorhanden.

4.2 Soll-Zustand

Der Soll-Zustand wurde definiert als ein fertiges Softwareprojekt, welche die Funktionalitäten in einem Client-, sowie einer Broker-Komponente beinhaltet.

Mangelnd fehlender Zeit in der Realisierungsphase konnten folgende Funktionalitäten während der Studienarbeitsdauer nicht abgedeckt werden:

- voll funktionsfähiger Client
  - Datenhaltung
  - Konfigurationsverwaltung
- Einsatz des Storage Commitment für Bereinigung des lokalen Zwischenspeichers
- Broker-Komponente
- Integration der Lösung in einen Druckertreiber

4.3 Weiterführende Arbeiten

Da diese Arbeit aus Zeitlichen- und Ressource-Gründen den Soll-Zustand nicht erreichen konnte, wäre ein Vervollständigen dieser angefangenen Komponenten denkbar.

Des Weiteren wäre in einer weiterführenden Arbeit die Funktionalität zum Importieren einer Bilddateien über einen Druckertreiber möglich, so dass jegliche Druckausgaben einer bestimmten Applikation alternativ an ein DICOM-Archiv umgeleitet werden können.

5 Persönlicher Bericht

Mit dieser Arbeit nahm ich die zweite und letzte Studienarbeit in Angriff, welche es innerhalb des FH-Diplom-Informatik Studiengangs an der HSR zu schreiben galt.

Nachdem ich in der ersten Studienarbeit in einem Dreier-Team gearbeitet hatte, war ich für diese Studienarbeit auf mich alleine gestellt. Die bis dahin gewonnen Erfahrungen aus der ersten Arbeit kamen mir beim Planen und Durchführen dieser Arbeit in gewissen Teilen zu Gute und ich versuchte dies noch effizienter zu gestalten.

Da das Thema in einen total anderen Bereich ging und für mich absolutes Neuland darstellte, brauchte ich viel Zeit um mich in diese Thematik einzuarbeiten. Durch den Einsatz unterschiedlicher Technologien und verschiedenen Komponenten galt es die fertige Lösung so zu entwickeln, dass diese möglichst einfach in eine bestehende Medizin-Infrastruktur einzusetzen ist. Ich strukturierte meine Arbeit so, dass auch bei eigenständiges Arbeiten jederzeit die Arbeit in einem Team möglich wäre und die Komponenten in sich eigenständige Projekte sind.

Für die Realisierung stand ein begrenztes Zeitbudget zur Verfügung, in welchem zu erst ein Prototyp entwickelt wurde, dieser im späteren Verlauf dann ständig erweitert wurde.

Ich bekam zu spüren, dass mit mir als einziges Teammitglied die Arbeitszeit zum Programmieren knapp bemessen ist und die Zeitabschätzungen bezüglich Realisierung der Softwarelösung zu optimistisch war und darum die gesetzten Soll-Kriterien nicht eingehalten werden konnten. Es ist sehr schwierig etwas abzuschätzen, wenn man das Arbeitsthema mit seinen Eigenheiten im Voraus nicht kennt. Im Nachhinein würde ich in der Projekt-Planung mehr Reserven einplanen, bzw. in der Realisierung des Softwareprojekt die Komponenten komponentenweise realisieren und in Betrieb nehmen.

Ein weiterer nicht zu unterschätzender Aufwand bestand darin die Dokumentation dieser Studienarbeit sicherzustellen. In den letzten 3 Wochen war ich nur noch am Verfassen der Dokumentation, da dies für eine erfolgreiche Abgabe notwendig ist.

Abschliessend kann gesagt werden, dass mich diese Studienarbeit sehr gefordert hat. Ich war für alles selbst verantwortlich und musste mich deshalb auch in Gebieten zurecht finden, in welchen ich nicht so stark bin (Java GUI-Design/Programmierung), bzw. noch keine Erfahrungen sammeln konnte (DICOM-Standard).

Trotzdem habe ich von dieser zweiten Studienarbeit extrem profitieren können und mir neues Wissen und Erkenntnisse angeeignet, was mich freut und auch mit einem gewissen Stolz erfüllt. Ich fand dieses Themengebiet spannend und es gab mir Einblicke in einen bisher unbekannten Bereich der Informatik, in welchen ich zukünftig bestimmt mehr Augenmerk darauf werfen werde. Nur finde ich es schade, dass das Projekt nicht wie anfangs geplant fertig abgeschlossen werden konnte, aber einen guten Stand aufweist, welche für Nachfolgearbeiten genutzt werden kann.

6 Dank

Ich möchte mich bei meinem Betreuer Prof. Dr. Axel Doering für die stets angenehme und wertvolle Zusammenarbeit bedanken. Durch seine praxisnahen Erfahrungen auf dem Gebiet der Medizin-Informatik und dem Software-Engineering konnte ich wertvolle Erkenntnisse gewinnen, welche mir für weitere Arbeiten bestimmt zugute kommen werden.

Ein weiterer Dank geht an meinen Kollegen und ex. Kommilitone Marco Facetti, welcher mir bei spezifischen Fragen zum Erstellen des Softwareprojekts behilflich war, oder auch beim Einholen einer Zweitmeinung zu Hilfe stand.

Zum Schluss danke ich all denjenigen, welche mir während dieser Durchführungszeit zur Seite gestanden sind und mich in unterschiedlichen Belangen unterstützt hatten.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

DICOM Basics

19. Mai 2009

Abstract

Dieses Dokument gibt einen Einblick in die Welt des DICOM Standards, welche die Basis dieser Studienarbeit bildet.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	DICOM_Basics_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
19.05.2009	Dokument erstellt
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument gibt einen Einblick in die Welt des DICOM Standards, welche die Basis dieser Studienarbeit bildet.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

In diesem Dokument wird der DICOM-Standard kurz erklärt, die eingesetzten Software für DICOM aufgelistet, danach der DICOM-Server erklärt, der Prozess der DICOM-Umwandlung beschrieben und zum Schluss ein Anwendungsbeispiel des dcm4che2 Toolkits aufgezeigt.

2 DICOM-Standard kurz erklärt

Digital Imaging and Communications in Medicine (DICOM) ist ein offener Standard zum Austausch von Informationen in der Medizin.

Diese Informationen können beispielsweise digitale Bilder, Zusatzinformationen wie Segmentierungen, Oberflächendefinitionen oder Bildregistrierungen sein.

DICOM standardisiert sowohl das Format zur Speicherung der Daten, als auch das Kommunikationsprotokoll zu deren Austausch.

Fast alle Hersteller bildgebender oder bildverarbeitender Systeme in der Medizin wie z. B. Digitales Röntgen, Magnetresonanztomographie, Computertomographie oder Sonografie implementieren den DICOM-Standard in ihren Produkten. Dadurch wird im klinischen Umfeld Interoperabilität zwischen Systemen verschiedener Hersteller erreicht.

DICOM ist auch die Grundlage für die digitale Bildarchivierung in Praxen und Krankenhäusern (Picture Archiving and Communication System, PACS).

Quelle: de.wikipedia.org, Artikel - Digital Imaging and Communications in Medicine http://de.wikipedia.org/wiki/Digital_Imaging_and_Communications_in_Medicine [Abruf: 2009-05-25]

2.1 Der DICOM-Standard

Der DICOM Standard wurde von der National Electrical Manufactures Association[1] (NEMA) erstellt, welche über unten stehenden Link zum herunterladen erhältlich ist (Entwürfe des Standards sind nach entsprechender Jahrzahl organisiert).

ftp://medical.nema.org/medical/dicom/ [Abruf: 2009-05-2005]

2.2 SCP-SCU

In der DICOM-Welt trifft man häufig auf die beiden Begriffe SCP, bzw. SCU, welche paarweise zueinander in Beziehung stehen.

Die Abkürzung SCP steht dabei für Service Class Provider, also ein Gerät/Applikation welches einen Dienst anbietet, wohingegen SCU (Service Class User) ein Gerät/Applikation bezeichnet, welche einen Dienst von einem SCP in Anspruch nimmt.

2.3 DICOM information model

DICOM verwendet eine eigene Sprache, basierend auf seinem Modell der realen Welt (model of the real world – DICOM information model).

Alle aus unseren realen Welt bekannten Daten (Patient, Studien, Medizinische Geräte usw.) sind durch DICOM als Objekte mit entsprechenden Eigenschaften oder Attribute festgehalten. Diese Definition dieser Objekte und Attribute wurden standardisiert entsprechend den DICOM Information Object Definition (IODs).

IODs können als eine Sammlung von Attribute bezeichnet werden, welches jedes einzelne Daten Objekt beschreibt. Ein Patienten IOD kann zum Beispiel einen Namen, Medizinische Patienten Nr (ID), Geschlecht, Alter, Gewicht usw. besitzen.

1 http://medical.nema.org/

2.4 IOD

IOD (Information Object Definition) repräsentieren Informationsobjekt der (realen) medizinischen Welt (z. B. Patient, Study, Series, Image) und deren Beziehungen untereinander.

2.4.1 Secondary Capture IOD

Beispiel einer IOD wäre Secondary Capture. Diese IOD wurde in DICOM vor ein paar Jahren hineingebracht, aus dem einfachen Grund, da digitale Bild-Daten nicht nur von Modalitäten (medizinisch für bildgebende Arbeitsstationen oder bilderfassenden Gerät) erstellt werden können, sondern auch von anderen Quellen (z. B. Einscannen von Filmdaten, Scannen von Berichte in Textform oder sogar Erstellen eines Screenshots auf der Arbeitsstation).

2.5 DICOM AE und AET

Der DICOM AE (AE = Application Entity) gehört generell zu irgend einer DICOM Applikation eines vernetzten DICOM Geräts, beispielsweise einem DICOM-Server (mit DICOM-Archiv), einer bildgebenden Arbeitsstation oder einem bilderfassenden Gerät (Modalität). Die Aufzählung deckt ca. 95% aller AEs ab, welche man in einer klinischen Umgebung antreffen wird.

Da sich jedes Gerät in einem Netzwerk befindet, wird vorausgesetzt, dass eine Netzwerkkarte vorhanden, bzw. diese mit einer eindeutige IP-Adresse versehen ist, damit die Geräte sich untereinander finden und kommunizieren können.

Als Ergänzung zur Netzwerkeinstellung fügt DICOM jedem AE seinen eigenen DICOM-Namen zu, welcher als „Application Entity Title“ (AET) bezeichnet wird Dieser Name darf per Definition nicht länger als 16 Zeichen lang sein. Ein gültiger DICOM AET wäre z. B. „PACSSERVER“.

3 DICOM-Software

Die nachfolgend aufgelisteten Softwarelösungen/Komponenten wurden für die Bearbeitung der DICOM spezifischen Aufgaben eingesetzt.

3.1 dcm4che

Dcm4che[2] ist eine Sammlung von Open Source Anwendungen und Hilfsmittel (utilities) für die Gesundheitsindustrie. Diese Anwendungen wurden alle in Java entwickelt und lassen sich dadurch einfach auf andere Systeme portieren, da Java plattformunabhängig ist. Voraussetzung für den Betrieb ist ein JDK in der Version 1.4 oder höher.

Der Kern des dcm4che Projektes ist eine robuste Implementierung des DICOM-Standards.

3.1.1 dcm4chee

Dcm4chee - das zusätzliche 'e' im Namen steht für 'enterprise' (Unternehmung) - ist ein Bild-Verwalter / Bild-Archiv gemäss IHE. Die Abkürzung IHE (Integrating the Healthcare Enterprise) steht dabei für eine Initiative von Anwendern und Herstellern, welche das Ziel haben den Informationsaustausch zwischen IT-Systemen im Gesundheitswesen zu standardisieren und zu harmonisieren.

Die Applikation enthält die DICOM, HL7 Services und Schnittstellen welche u.a. Speicher, Abfrage und Arbeitsablauf in einer Gesundheitsumgebung unterstützen. HL7 (Health Level 7) ist eine Gruppe internationaler Standards, welche den Austausch von Daten zwischen Organisationen im Gesundheitswesen und deren Computersystemen regelt.

dcm4chee ist vor verpackt und verteilt innerhalb dem JBoss Application-Server. Durch den Einsatz vieler JBoss Funktionen (JMS, EJB, Servlet Engine, etc.) und im Bezug auf unterschiedlichen Rollen gemäss IHE, bietet die Anwendung viele robuste und skalierbare Services.

3.1.2 dcm4che2 Toolkit

Das Toolkit ist eine nützliche Sammlung von kleinen Programmen, welche spezifische Funktionalitäten zur Verfügung stellen.

Es befindet sich in der aktuellsten Version 2.x, welche im Bezug auf hohe Leistung und Flexibilität neu konzipiert wurde.

Das Toolkit kann auf der Webseite von dcm4che unter dem Bereich Download heruntergeladen werden. Man wird danach auf eine Downloadseite von SOURCEFORGE.NET weitergeleitet.

3.2 DICOM viewer

Mittels DICOM viewer können die erstellten DICOM Dateien angeschaut werden.

Folgende Produkte sind gemäss dcm4che.orgzu empfehlen:

http://www.dcm4che.org/confluence/display/ee2/DICOM+Viewers [Abruf: 2009-05-25]

2 http://www.dcm4che.org/

- Osirix, http://www.osirix-viewer.com/Downloads.html (nur für MacOS X)
- K-PACS, http://www.k-pacs.de (für Windows XP/2000)
- Clear Canvas Workstation, http://www.clearcanvas.ca
- Aeskulap, http://aeskulap.nongnu.org/ (für Windows und Linux)

4 DICOM-Server

Unter dem Begriff DICOM-Server wird eine Softwarelösung bezeichnet, welche den DICOM-Standard implementiert und diverse SC (Service Class) anbietet. Als Service Class werden Dienstklassen benennt, welche bestimmte Aktionen ausführen können (z. B. Store, Print, Query, Retrieval, Modality Worklist, Storage Commitment usw.)

Werden diese Dienstklassen durch einen Client benutzt, so fungiert der Client dabei als SCP (Service Class User), bzw. der DICOM-Server als SCP (Service Class Provider).

Diese Softwarelösung wird auf einer entsprechenden physikalischen Hardware, heute vermehrt auch als virtueller Server in einer Virtuellen Serverinfrastruktur (dies ist aber keine Voraussetzung zum Einsatz), in der Medizin-Informatik betrieben.

Der DICOM-Server ist die zentrale Instanz im Netzwerk einer Medizinalumgebung, welcher entsprechende Schnittstellenfunktionalitäten seinen Clients anbietet.

Ein DICOM-Server könnte beispielsweise jener aus der dcm4che.org Open Source Lösung sein.

Damit dieser Server benutzt werden kann, müssen folgende Informationen bekannt sein:

- Netzwerkinformationen
  - IP-Adresse, Port
- DICOM-Informationen
  - DICOM AET, z. B. „PACSSERVER“ (siehe Abschnitt: 2.5 DICOM AE und AET)

Für diese Studienarbeit wurde der DICOM-Servers aus dem dcm4che.org Projekt verwendet. Für die weitere Informationen zur Installation und Konfiguration des DICOM-Servers wird auf das Dokument *Betreiberdokumentation* verwiesen.

5 Vorgang Umwandlung in ein DICOM Format

Damit die zu sendenden Daten auf dem DICOM-Server entgegengenommen und archiviert werden können, müssen die Daten zuerst entsprechend umgewandelt werden.

Anhand des Dateiformats sollte durch eine gegebene Applikationslogik die entsprechende DICOM SOP Class verwenden werden, welche die Daten in ein entsprechendes DICOM-Objekt konvertieren/kapselt, das danach die DICOM Konformität erfüllt.

5.1 DICOM SOP Classes

Diese Tabelle listet die jeweiligen SOP Klassen auf, welche je nach gegebenem Dateiformat für die Kapselung der Ressource in ein DICOM-Objekt verwendet werden müssen:

Name	**SOP Class UID**[3]	**für Dateiformat**
Secondary Capture Image Storage	1.2.840.10008.5.1.4.1.1.7	JPEG
Encapsulated PDF Storage	1.2.840.10008.5.1.4.1.1.104.1	PDF

Table 1: DICOM SOP Classes

5.2 DICOM SOP Instance UID

Jedes erstellte DICOM-Objekt wird mit einer entsprechenden DICOM SOP Instance UID gekennzeichnet. Diese wird anhand des eingesetzten DICOM-Toolkit (dcm4che-2.0.x) aus folgenden Komponenten zusammengesetzt und automatisch generiert.

1.2.40.0.13.1.<host-ip>.<zahlenfolge>.<uhrzeit>

Beispiel: 1.2.40.0.13.1.127.0.0.1.900409598.20090422231915627.3

Innerhalb der DICOM-Welt sind die DICOM UID Root eindeutig definiert und müssen bei Registrierungsstellen entsprechend registriert werden.

http://www.dclunie.com/medical-image-faq/html/part8.html#UIDRegistration [Abruf: 2009-05-25]

5.2.1 Beispiel dcm4che.org

Implementation Class UID	1.2.40.0.13.1.1
Implementation Version Name	dcm4che-2.0.19

Diese Attribute sind da um zu kennzeichnen, wer oder woher man in der DICOM-Welt kommuniziert. Die Implementation Class UID ist fix, hingegen kann der Implementation Version Name frei gewählt werden, sollte aber das eingesetzte DICOM-Toolkit identifizieren.

5.2.2 Beispiel Jpeg-Bilddatei

Ist ein Jpeg-Datei zu konvertieren, dann muss die DICOM SOP Secondary Capture verwendet werden. Die DICOM SOP Secondary Capture besagt, dass dieses Bild durch ein anderes Bildgerät als ein Computer Tomograph (CT) erstellt wurde.

3 Unique Identifier

5.3 DICOM-Objekt

Wird erstellt aus Dateien (z. B. externen importierten Bilder wie *.jpeg, *.bmp, *.tif) und dessen Meta-Informationen (Patientendaten etc.)

Nachfolgend eine Aufstellung aller Attribute, welche bei einem DICOM-Objekt während einem Datei-Import gesetzt werden:

```
(0008,0018) : SOP Instance UID
(0020,000D) : Study Instance UID
(0020,000E) : Series Instance UID
(0028,0002) : Samples per Pixel
(0028,0004) : Photometric Interpretation
(0028,0006) : Planar Configuration
(0028,0010) : Rows
(0028,0011) : Columns
(0028,0100) : Bits Allocated
(0028,0101) : Bits Stored
(0028,0102) : High Bit
(0028,0103) : Pixel Representation
(0028,1050) : Window Center
(0028,1051) : Window Width
(7FE0,0010) : Pixel Data
```

Bemerkung: alle Gruppen und Element-Nummern sind im Hexadezimal-Format definiert, Study Instance UID, Series Instance UID sind beide unique.

5.3.1 Type 1 Attribute

Sind Attribute, welche gesetzt werden müssen und einen gültigen Wert besitzen, bevor ein DICOM-Objekt erstellt werden kann:

```
(0008,0016) : SOP Class UID   : 1.2.840.10008.5.1.4.1.1.7
(0008,0064) : Conversion Type :
                                DV = Digitized Video
                                DI = Digital Interface
                                DF = Digitized Film
                                WSD = Workstation
                                SD = Scanned Document
                                SI = Scanned Image
                                DRW = Drawing
                                SYN = Synthetic Image
```

5.3.2 Type 2 Attribute

Sind Attribute, welche vorhanden sein müssen, aber einen Null-Wert besitzen dürfen:

```
(0010,0010) : Patient Name
(0010,0020) : Patient ID
(0010,0030) : Patient Date of Birth
(0010,0040) : Patient Sex
(0008,0020) : Study Date
(0008,0030) : Study Time
(0008,0050) : Accession Number
(0008,0090) : Referring Physician's Name
(0020,0010) : Study ID
(0020,0011) : Series Number
(0020,0013) : Instance Number
```

```
(0020,0020) : Patient Orientation
```

Es gibt noch Type 3 Attribute, welche vollständig fehlen können.

Für weitere Infos siehe:

http://www.medicalconnections.co.uk/wiki/How_to_make_DICOM_files_from_external_Images [Abruf: 2009-05-22-25]

6 Anwendungsbeispiel des dcm4che2 Toolkits

Dieses Kapitel zeigt ein konkretes Anwendungsbeispiel des dcm4che2 Toolkits auf.

6.1 dcm4che2 Toolkit

Die aktuellste Version des Package dcm4che2 ist der Release 2.0.19, welcher am 6. April, 2009 veröffentlicht wurde.

Folgende Dateien sind herunterzuladen:

- dcm4che-2.0.19-apidocs.zip
- dcm4che-2.0.19-bin.zip
- dcm4che-2.0.19-src.zip

Die Zip-Archive sind entsprechend zu entpacken und in einen zentralen Ordner im Dateisystem des Clients abzulegen.

Ich habe das Toolkit ins Verzeichnis /opt/dcm4che2/ auf meiner Arbeitsstation gespeichert.

/opt/dcm4che2/dcm4che-2.0.19
/opt/dcm4che2/dcm4che-2.0.19-apidocs

6.2 Szenario: Manueller Import einer Ressource an den DICOM-Server

Voraussetzungen:

- lauffähiger DICOM-Server ist in Betrieb
- dcm4che2 DICOM Toolkit ist vorhanden und einsatzbereit
- Ressource ist vorhanden, welche man auf den DICOM-Server importieren möchte (z. B. Jpeg-Datei -> siehe: homer.jpg)
- Eine Beschreibung der Ressource ist vorhanden (ist eine Textdatei, welche diverse Attribute besitzt -> siehe: homer.cfg)

Beispieldateien sind innerhalb des dcm4che2 Toolkits zu finden, siehe Verzeichnis:

/opt/dcm4che2/dcm4che-2.0.19/dcm4che-tool/dcm4che-tool-jpg2dcm/src/main/config

Datei: homer.jpg und homer.cfg

Beispiel einer ausführlichen Konfigurationsdatei:

/opt/dcm4che2/dcm4che-2.0.19/dcm4che-tool/dcm4che-tool-jpg2dcm/src/main/resources/org/dcm4che2/tool/jpg2dcm/jpg2dcm.cfg

siehe: http://www.dcm4che.org/confluence/display/d2/dcm4che2+DICOM+Toolkit [Abruf: 2009-05-25]

6.2.1 Umwandeln des Bildes in eine DICOM-Datei

Ins Verzeichnis des Toolkits wechseln:

```
$ cd /opt/dcm4che2/dcm4che-2.0.19/bin
```

Mit dem nachfolgenden Befehl wird eine Jpeg-Datei homer.jpg mit der dazugehörigen Konfigurationsdatei homer.cfg in eine DICOM-Datei homer.dcm konvertiert.

```
$ ./jpg2dcm -c /home/fvetter/Pictures/DicomTest/homer.cfg
/home/fvetter/Pictures/DicomTest/homer.jpg
/home/fvetter/Pictures/DicomTest/homer.dcm
```

Dieses Tool kapselt die Jpeg-Datei zusammen mit den Konfigurationsinformationen in eine DICOM-Datei.

Ausgabe von Konsole:

```
Encapsulated /home/fvetter/Pictures/DicomTest/homer.jpg to
/home/fvetter/Pictures/DicomTest/homer.dcm in 267ms.
```

Datei	homer.cfg	homer.jpg
Inhalt	# Patient Module Attributes # Patient's Name 00100010:Simpson^Homer^Jay # Patient ID 00100020:7431 # Patient's Birth Date 00100030:19551005 # Patient's Sex 00100040:M	

Table 2: Inhalt der zum konvertierenden Dateien (homer.cfg, homer.jpg)

6.2.2 DICOM-Datei an DICOM-Server senden

In diesem Abschnitt wird die vorher erstellte DICOM-Datei an den DICOM-Server gesendet.

Diesmal wird das Tool dcmsnd verwendet, welches auch wieder im dcm4che2 DICOM Toolkit zu finden ist.

Mit dem nachfolgenden Befehl wird eine DICOM-Datei homer.dcm an den DICOM-Server gesendet. Dieser besitzt die IP Adresse 192.168.168.11 und hört auf dem Port 11112 auf eingehende Verbindungen. Damit der Server genutzt werden kann muss vorher noch ein Benutzer „DCM4CHEE“ mitgegeben werden.

```
$ ./dcmsnd DCM4CHEE@192.168.168.11:11112 /home/fvetter/Pictures/DicomTest/homer.dcm
```

Ausgabe von Konsole:

```
Scanning files to send

.

Scanned 1 files in 0.036s (=36ms/file)

14:22:59,443 INFO  - Association(1) initiated
Socket[addr=/192.168.168.11,port=11112,localport=56664]

14:22:59,448 INFO  - DCM4CHEE(1): A-ASSOCIATE-RQ DCM4CHEE << DCMSND
```

```
14:22:59,453 INFO   - DCM4CHEE(1): A-ASSOCIATE-AC DCMSND >> DCM4CHEE

Connected to DCM4CHEE@192.168.168.11:11112 in 0.031s

14:22:59,474 INFO   - DCM4CHEE(1) << 1:C-STORE-RQ[pcid=3, prior=0

       cuid=1.2.840.10008.5.1.4.1.1.7/Secondary Capture Image Storage

       iuid=1.2.40.0.13.1.127.0.0.1.900409598.20090429142227960.3

       ts=1.2.840.10008.1.2.4.50/JPEG Baseline (Process 1)]

14:22:59,594 INFO   - DCM4CHEE(1) >> 1:C-STORE-RSP[pcid=3, status=0H]

.

Sent 1 objects (=45.05664KB) in 0.142s (=317.30026KB/s)

14:22:59,595 INFO   - DCM4CHEE(1) << A-RELEASE-RQ

14:22:59,597 INFO   - DCM4CHEE(1) >> A-RELEASE-RP

Released connection to DCM4CHEE@192.168.168.11:11112

14:22:59,598 INFO   - DCM4CHEE(1): close
Socket[addr=/192.168.168.11,port=11112,localport=56664]
```

6.2.2.1 Storage Commitment

Durch das Anfügen der Option -stgcmt kann ein Storage Commitment verlangt werden.

Für eine Übersicht aller zusätzlichen Optionen zum Tool dcmsnd, wird auf folgenden Link verwiesen: http://www.dcm4che.org/confluence/display/d2/dcmsnd [Abruf: 2009-05-22]

```
 ./dcmsnd -stgcmt DCM4CHEE@192.168.168.11:11112
/home/fvetter/Pictures/DicomTest/homer.dcm

Scanning files to send

.

Scanned 1 files in 0.101s (=101ms/file)

21:32:18,905 INFO   - Association(1) initiated
Socket[addr=/192.168.168.11,port=11112,localport=46856]

21:32:18,909 INFO   - DCM4CHEE(1): A-ASSOCIATE-RQ DCM4CHEE << DCMSND

21:32:18,914 INFO   - DCM4CHEE(1): A-ASSOCIATE-AC DCMSND >> DCM4CHEE

Connected to DCM4CHEE@192.168.168.11:11112 in 0.031s

21:32:18,952 INFO   - DCM4CHEE(1) << 1:C-STORE-RQ[pcid=5, prior=0

       cuid=1.2.840.10008.5.1.4.1.1.7/Secondary Capture Image Storage

       iuid=1.2.40.0.13.1.127.0.0.1.900409598.20090429142227960.3

       ts=1.2.840.10008.1.2.4.50/JPEG Baseline (Process 1)]

21:32:18,983 INFO   - DCM4CHEE(1) >> 1:C-STORE-RSP[pcid=5, status=0H]

.

Sent 1 objects (=45.05664KB) in 0.068s (=662.5976KB/s)

21:32:18,987 INFO   - DCM4CHEE(1) << 2:N-ACTION-RQ[pcid=1, actionID=1

       cuid=1.2.840.10008.1.20.1/Storage Commitment Push Model SOP Class

       iuid=1.2.840.10008.1.20.1.1
```

```
       ts=1.2.840.10008.1.2/Implicit VR Little Endian]

21:32:19,022 INFO   - DCM4CHEE(1) >> 2:N-ACTION-RSP[pcid=1, actionID=null,
status=a801H

       error=DCMSND]

WARNING: Storage Commitment request failed with status: A801H

(0000,0002) UI #20 [1.2.840.10008.1.20.1] Affected SOP Class UID

(0000,0100) US #2 [33072] Command Field

(0000,0120) US #2 [2] Message ID Being Responded To

(0000,0800) US #2 [257] Data Set Type

(0000,0900) US #2 [43009] Status

(0000,0902) LO #6 [DCMSND] Error Comment

(0000,1000) UI #22 [1.2.840.10008.1.20.1.1] Affected SOP Instance UID



21:32:19,056 INFO   - DCM4CHEE(1) << A-RELEASE-RQ

21:32:19,057 INFO   - DCM4CHEE(1) >> A-RELEASE-RP

21:32:19,057 INFO   - DCM4CHEE(1): close
Socket[addr=/192.168.168.11,port=11112,localport=46856]

Released connection to DCM4CHEE@192.168.168.11:11112
```

In diesem Beispiel führte das Storage Commitment zu einem Fehler.

Da diese Fehlermeldung „Storage Commitment request failed with status: A801H“ nicht aussagekräftig ist, wurde via Internetrecherche mehr Informationen zum Fehlercode A801H gesucht.

Unter der URL http://www.pacsone.net/conformance.htm [Abruf: 2009-05-25] kam heraus, dass der Fehler Code A801H für „Refused Move destination unknown“ steht. Diese Meldung ist nach wie vor unverständlich, könnte vermutlich etwas mit der Konfiguration des DICOM-Servers zu tun haben.
Anmerkung: Die Lösung zu diesem Problem konnte während der Studienarbeit nicht mehr herausgefunden werden.

6.2.3 Überprüfen der gesendeten Daten

Hat der Import der DICOM-Datei geklappt, so befindet sich diese auf dem DICOM-Server. Überprüft werden kann dies, wenn man sich mittels Web-GUI auf den Server verbindet und eine Abfrage auf die DICOM-Datenbank macht.

In dem vorigen Beispiel kann nach dem Patienten Name „Simpson^Homer^Jay, oder via seiner persönlichen Patienten ID 7431 gesucht werden.

Die nachfolgende Abbildung zeigt den Inhalt (DICOM-dataset) der importierten DICOM-Datei an, welche nebst dem eigentlichen Bild auch die Metadaten (Patienten- und Untersuchungsdaten) beinhaltet.

DICOM dataset

Group	Element	Name	VR	Length	VM	Value
0008	0005	Specific Character Set	CS	10	1	ISO_IR 100
0008	0012	Instance Creation Date	DA	8	1	20090422
0008	0013	Instance Creation Time	TM	10	1	231915.628
0008	0016	SOP Class UID	UI	26	1	1.2.840.10008.5.1.4.1.1.7
0008	0018	SOP Instance UID	UI	54	1	1.2.40.0.13.1.127.0.0.1.900409598.20090422231915627.3
0008	0020	Study Date	DA	0	0	
0008	0030	Study Time	TM	0	0	
0008	0050	Accession Number	SH	0	0	
0008	0054	Retrieve AE Title	AE	8	1	DCM4CHEE
0008	0056	Instance Availability	CS	6	1	ONLINE
0008	0060	Modality	CS	2	1	OT
0008	0061	Modalities in Study	CS	2	1	OT
0008	0064	Conversion Type	CS	2	1	SI
0008	0070	Manufacturer	LO	0	0	
0008	0090	Referring Physician's Name	PN	0	0	
0010	0010	Patient's Name	PN	18	1	Simpson^Homer^Jay
0010	0020	Patient ID	LO	4	1	7431
0010	0030	Patient's Birth Date	DA	8	1	19551005
0010	0040	Patient's Sex	CS	2	1	M
0020	000D	Study Instance UID	UI	54	1	1.2.40.0.13.1.127.0.0.1.900409598.20090422231915627.1
0020	000E	Series Instance UID	UI	54	1	1.2.40.0.13.1.127.0.0.1.900409598.20090422231915627.2
0020	0010	Study ID	SH	0	0	
0020	0011	Series Number	IS	2	1	1
0020	0013	Instance Number	IS	2	1	1
0020	1206	Number of Study Related Series	IS	1	1	1
0020	1208	Number of Study Related Instances	IS	1	1	1
0020	1209	Number of Series Related Instances	IS	1	1	1
0028	0002	Samples per Pixel	US	2	1	3
0028	0004	Photometric Interpretation	CS	12	1	YBR_FULL_422
0028	0006	Planar Configuration	US	2	1	0
0028	0010	Rows	US	2	1	585
0028	0011	Columns	US	2	1	432
0028	0100	Bits Allocated	US	2	1	8
0028	0101	Bits Stored	US	2	1	8
0028	0102	High Bit	US	2	1	7
0028	0103	Pixel Representation	US	2	1	0
0032	000A	Study Status ID	CS	0	0	
0043	0010		LO	15	1	dcm4che/archive
0043	1010		OB	8	1	0\0\0\0\0\0\0\6
0043	1011		OB	8	1	0\0\0\0\0\0\0\4
0043	1012		OB	8	1	0\0\0\0\0\0\0\4
0043	1013		OB	8	1	0\0\0\0\0\0\0\4
0043	1014		AE	6	1	DCMSND
0088	0130	Storage Media File-set ID	SH	0	0	
0088	0140	Storage Media File-set UID	UI	0	0	

Illustration 1: DICOM-dataset

6.2.3.1 Erläuterung

Die Study Instance UID wurde vom System (hier dem Utility-Tool dcmsnd) vorgegeben. Wichtig ist dabei, dass diese UID unique ist.

Auch ist sehr schön ersichtlich, dass dieses Bild via der SOP Class UID klar als Secondary Capture Bild gekennzeichnet ist, sowie die zusätzlich mitgegebenen Attribute (z. B. Patient's Birth Date, Patient's Sex) sind erkennbar.Die zu empfangende AE war in diesem Beispiel der DICOM-Server „DCM4CHEE“.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Anforderungsanalyse

19. Mai 2009

Abstract

Dieses Dokument beinhaltet die Anforderungsanalyse dieser Studienarbeit.

Inhalt dabei sind Ist-/Soll-Zustand mit funktionalen- und nicht funktionalen Anforderungen, Szenarios, Use Cases, Domain Model und Aktivitätsdiagramm.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Anforderungsanalyse_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
05.03.2009	Dokument erstellt
19.05.2009	Dokument in Version 2 überarbeitet
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument beschreibt die Anforderungsanalyse zur Studienarbeit.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Inhalt sind Ist-/Soll-Zustand mit funktionalen- und nicht funktionalen Anforderungen, Szenarios, Use Cases, Domain Model und Aktivitätsdiagramm.

2 Arbeitsauftrag

Der Arbeitsauftrag wurde durch den Verantwortlichen der Studienarbeit geschrieben und danach im hochschuleigenen Arbeits Verwaltungs Tool[1] (AVT) online publiziert.

Die Studenten hatten sich danach in einer Bewerbungsphase für die ausgeschriebenen Arbeiten zu bewerben. Zu einem späteren Zeitpunkt wurde dann die jeweiligen Arbeiten den Studenten zugeteilt.

Nachfolgend die Ausschreibungsinformationen zu dieser Studienarbeit:

2.1 Thema

Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur

2.2 Organisation

Studiengang:	Informatik (I)
Semester:	FS 2009 (16.02.2009-13.09.2009)
Durchführung:	Studienarbeit
Fachrichtung:	Software
Institut:	IFS: Institut für Software
Gruppengrösse:	1-2 Studierende
Status:	zugewiesen
Verantwortlicher:	Doering, Axel
Betreuer:	Doering, Axel
Gegenleser:	[Nicht definiert]
Experte:	[Nicht definiert]
Industriepartner:	[Nicht definiert]

Table 1: Organisation Studienarbeit

2.3 Ausschreibung

Anbei der Ausschreibungstext dieser Studienarbeit:

1 http://avt.hsr.ch/

Nur als Studienarbeit geeignet:

Ausgangslage, Problembeschreibung

DICOM (Digital Imaging and Communication in Medicine) und HL7 (Health Level 7) sind wichtige Standards zum Austausch von Informationen zwischen (meist bildgebenden) Medizin- und Laborgeräten und zentralen Informationssystemen (z.B. einem Klinik-Informationssystem KIS oder einem Labor-Informationssystem LIS). In zunehmendem Masse werden Patientendaten nicht mehr an einzelnen Geräten eingetragen, sondern über einen Worklist-Mechanismus von einem führenden System abgefragt. In der Gegenrichtung werden Bilder, andere Diagnosedaten und abrechnungsrelevante Daten (z.B. über durchgeführte Untersuchungen) an verschiedene zentrale Systeme übertragen. Durch eine konsequente Vernetzung aller beteiligten Instanzen und entsprechende Anpassung des Arbeitsablaufs ist eine Steigerung der Gesamteffizienz um ca. 20-30% nachweisbar möglich.
Ein wesentliches Problem bei der Einführung eines elektronischen Workflows ist jedoch die Inkompatibilität vorhandener älterer Geräte ohne standardisierte Schnittstellen. Für den Anwender ist die Ablösung dieser Geräte aufgrund der hohen Investitionskosten meist nicht ohne Weiteres möglich; zum Teil sind auch aktuelle Geräte mit gleicher Funktionalität noch nicht mit diesen Schnittstellen ausgestattet. Viele Geräte verfügen jedoch über eine Möglichkeit, Messdaten auszudrucken, die für den Zweck einer Anbindung an ein DICOM-Archiv verwendet werden könnte.

Aufgabenstellung

Ziel der Studienarbeit ist der Entwurf und die Entwicklung eines Client-Server-basierten Systems, mit dem Daten von Altgeräten ohne DICOM-Schnittstelle über die Verwendung der Druckfunktion in ein DICOM-Archiv eingespeist werden können. Der Client soll dabei als „Druckertreiber“ auf dem Gerät installiert werden und mit dem Server über ein geeignetes proprietäres Protokoll kommunizieren. Dabei ist auf die Sicherheit (Vertraulichkeit und Schutz gegen Verluste und Manipulation) der Datenübertragung zu achten. Serverseitig erfolgt die Umsetzung der Daten in das DICOM Protokoll und die Kommunikation mit einem (konfigurierbaren) DICOM Archiv. Im Minimum muss der Server die Funktionen Service Class User (SCU) Storage Visible Light IOD, SCU Storage Secondary Capture IOD und SCU Storage Encapsulated PDF IOD implementieren.
Funktionsweise des Client und das Protokoll müssen auf Unterbrechungen der Verbindung zwischen Client und Server ausgelegt werden. Beispielsweise könnte der Client einen lokalen Cache für übertragene Daten führen, der erst nach einer Quittierung der erfolgreichen Ablage im DICOM Archiv (entsprechende Mechanismen sind im DICOM Protokoll vorgesehen) bereinigt wird. Der Client wiederholt Übertragungsversuche bei ausbleibender Quittierung, die der Server zuordnen und bearbeiten muss.
Ferner ist ein Update-Mechanismus und eine zentrale Konfiguration für die Clients vom Server aus vorzusehen.

Zur Durchführung

Die Bearbeitung der Aufgabe setzt eine Einarbeitung in die Grundlagen des DICOM Standards voraus, für die ca. 15 Arbeitsstunden anzusetzen sind. Für die Implementierung der Server-Funktionalität kann auf Bibliotheken (z.B. JDICOM) zurückgegriffen werden. Für den Aufbau einer Testumgebung stehen frei verfügbare DICOM Server mit ausreichendem Funktionsumfang zur Verfügung (z.B. DCM4CHE, www.dcm4che.org). Erwartet wird der Aufbau eines Buildsystems für einen kontinuierlichen Buildvorgang, bei dem gleichfalls Unittests (ggf. auch Integrations- und Systemtests) durchgeführt werden sollen.
Mit dem Betreuer finden in der Regel wöchentliche Besprechungen statt. Zusätzliche

Besprechungen sind nach Bedarf durch die Studierenden zu veranlassen. Besprechungen mit dem Auftraggeber werden nach Bedarf durchgeführt. Alle Besprechungen sind von den Studenten mit einer Traktandenliste vorzubereiten und die Ergebnisse sind in einem Protokoll zu dokumentieren, das dem Betreuer per E-Mail zugestellt wird. Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. An Meilensteinen gemäss Projektplan sind einzelne Arbeitsresultate in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsresultate erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Implementierung und Dokumentation. Voraussetzungen: [Nicht definiert]

Table 2: Studienarbeit Ausschreibungstext

2.4 Bewerbungen

Gruppe:	Vetter
Status:	Arbeit zugewiesen (Priorität Student: 1)
Studierende:	- Vetter, Florian
Kommentar:	gemäss Angaben von Lothar Müller

Table 3: Studienarbeit Bewerbungsübersicht

3 Ist-Zustand

3.1 Ausgangslage

Im Medizinalbereich ist ein stetiger technologischer Wandel im Gange. Gerade Altgeräte, d.h. meist bildgebende Medizin- und Laborgeräte, welche heute noch vielerorts im produktiven Einsatz anzutreffen sind, können bezüglich Funktionsumfang und Kompatibilität nicht in aktuelle Systeme integriert werden. Es fehlt ihnen an einer entsprechenden Schnittstelle (DICOM-Schnittstelle).

Da die Lebensdauer dieser Geräte auf ca. 10 Jahre ausgelegt sind, können diese nicht so schnell durch neuere Modelle ausgetauscht werden, da dies mit hohen Anschaffungskosten verbunden sind.

Wenn man jene Geräte an ein DICOM Archiv anschliessen möchte, so ist das nicht ohne weiteres möglich. Man muss die entsprechende Funktionalität auf eine andere Art nachrüsten.

3.2 Möglicher Lösungsansatz

Laborgeräte sind meist mit einem Computer verbunden, welche ein Microsoft Windows als Betriebssystem verwendet. Eine Möglichkeit könnte darin bestehen einen virtuellen Drucker auf diesen Geräten zu installieren (Druckertreiber-Installation), welche die Möglichkeit bietet die Daten als Bilder/PDF-Datei zu exportieren. Dazu müssen diese Dateien zuerst aufbereitetet, d.h. in ein entsprechendes Format konvertiert und mit Metainformationen versehen werden, damit diese anschliessend an ein DICOM-Archiv weitergeleitet werden können.

Da die Schnittstellen der Medizingeräte softwarebasiert ist, könnte mit Hilfe entsprechender Software eine Integration in die DICOM-Welt sichergestellt werden.

4 Soll-Zustand

In diesem Abschnitt wird beschrieben, wie die zu entwickelnde Lösung aussehen soll. Man unterscheidet dabei zwischen funktionalen und nicht funktionalen Anforderungen. Die erstellte Lösung soll später in einem medizinischen Umfeld produktiv genutzt werden können.

4.1 Ziel

Ziel dieser Lösung ist es eine lokal vorhandene Ressource (z. B. eine Bilddatei) mit entsprechender Beschreibungen (Metainformationen) auf ein serverseitiges Archiv zu importieren. In diesem Prozess wird dabei sichergestellt, dass diese Ressource auf dem DICOM-Server in seinem Archiv vollständig archiviert wurde. Die Verantwortung der Daten gehen dabei vom Client an den DICOM-Server/DICOM-Archiv über.

4.2 Funktionale Anforderungen

Die zu entwickelnde Lösung muss folgende Eigenschaften erfüllen:

- DICOM Konformität erfüllen
- Stellt Datenaustausch von Altgeräte ohne DICOM-Schnittstelle via Druckfunktion in ein DICOM-Archiv sicher.
- Die Funktionalität des virtuellen Druckertreibers wird definiert durch:
  - Möglichkeit eine Bilddatei zu erstellen
  - Möglichkeit eine PDF Datei zu erstellen
  - Eingabe von Metainformationen zum Patienten/Studie
- Implementiert folgende Schnittstellen zur Anbindung an ein DICOM-Archiv:
  - SCU[2] Storage Visible Light IOD[3]
  - SCU Storage Secondary Capture IOD
  - SCU Storage Encapsulated PDF IOD
- Ferner ist ein Update-Mechanismus und eine zentrale Konfiguration für die Clients vom Server aus vorzusehen.
  - zentrale Konfiguration:
    - Server verwaltet die Clients mit ihren Konfigurationen für die Verbindung an ein DICOM-Archiv
  - Update-Mechanismus:
    - stellt sicher, dass die Clients aktualisiert werden können
      - Aktualisierung des Druckertreibers (binarys)
- Ist integrierbar in ein bestehendes Computer Netzwerk (TCP/IP basiert)

2 SCU = Service Class User
3 IOD = Information Object Definition

- Sichere Datenübertragung
  - Vertraulichkeit (Kommunikation kann verschlüsselt sein)
  - Schutz gegen Verluste und Manipulationen
- Besitzt folgende Verbindungseigenschaften:
  - Resistenz gegen Verbindungsunterbrüche zwischen Client, Broker und DICOM-Server
  - Client wiederholt Übertragungsversuche bei ausbleibender Quittierung durch den DICOM-Server
  - Client führt einen lokalen Cache/Queue für die zu Übertragung anstehenden Daten
  - Client bereinigt Cache erst dann, wenn eine Quittierung über einer erfolgreichen Ablage im DICOM-Archiv durch den DICOM-Server erhalten wurde.

4.3 nicht funktionale Anforderungen

Der Client (virtueller Druckertreiber) wird auf einem Computer installiert/eingerichtet, welcher mit einem Medizin- oder Laborgerät verbunden ist.

Die zu entwickelnde Lösung muss somit:

- Microsoft Windows XP als Plattform unterstützen
- Der virtuelle Druckertreiber (Printer driver) muss für die Windows Plattform (Microsoft Windows XP) programmiert werden

5 Szenarios

Folgender Abschnitt beschreibt mögliche Szenarien, welche in der Anforderungsanalyse zusammengetragen wurde. Die Szenarien zeigen Anforderungen des Benutzers und möglichen Interaktion mit dem System auf.

5.1 Szenario 1: Bild an DICOM Server senden

Der Hautarzt Dr. Skin ist eine angesehene Persönlichkeit und Fachmann in seinem praktizierenden Gebiet (Dermatlogie). Er ist ca. 58 jährig, ein Mann alter Schule und weit über die Landesgrenze hinaus bekannt für seine erfolgreichen Behandlungen, welche ihm viele Kunden im In- sowie Ausland bringen. Auch wird er als Berater für Expertenmeinungen zu besonders schwierigen Fälle herbeigezogen.

In seinem beruflichen Alltag gibt es oft Situationen, wo er zu Patienten auf Visite gehen muss, oder in andere Kliniken gerufen wird. Immer mit dabei hat er aber seine etwas in die Jahre gekommene Bildkamera, welche er für Aufnahmen der Hautoberfläche eines Patienten einsetzt. Sie leistet für diesen Zweck ihren Dienst, ist aber nicht mehr mit den neusten Geräte kompatibel.

Dr. Skin kennt die Vorzüge moderner Medizininformations-Systemen und möchte gerne seine anfallenden Daten als Sicherung in ein elektronisches Archiv kopieren.

Da dieses Gerät nicht DICOM kompatibel ist, verwendet er eine Lösung, welche ihm ermöglicht diese Daten über eine nachrüstbare Schnittstelle an ein DICOM Archiv zu senden. Dazu braucht er nur einen Computer (vorzugsweise einen Notebook) und ein Bildbetrachtungsprogramm, welche mit einem virtuellen Druckertreiber ausgerüstet ist.

Als erstes werden die erstellten Bilder via einem virtuellen Druckertreiber zum Import vorbereitet, d.h. der Treiber löst einen Druckdialog aus, welche den Anwender zur Eingabe patientenspezifischen Informationen (Name, Vorname, Patienten ID, Geburtsdatum, Geschlecht) sowie Informationen zur Studie (Untersuchungstyp, Datum der Untersuchung, Bildbeschreibungen) auffordert.

Das zum Import vorgesehene Bild wird zusammen mit den Patienten- und Bilder-Metainformationen lokal auf seinem Computer in einer Queue zwischengespeichert. Da Dr. Skin nicht im Spital ist, müssen die Daten lokal aufbewahrt werden, damit diese nicht verloren gehen.

Kommt er bei sich im Spital wieder an und verbindet sein Notebook ans Netzwerk des Spitals, werden automatisch diese Daten an das DICOM-Archiv gesendet. Bei erfolgreichem Importieren in das DICOM-Archiv, wird der Client die lokal gespeicherten Daten löschen.

Nun befindet sich die Daten mit allen relevanten Informationen zum Patienten/der Untersuchung in einem elektronischen System (Archiv) und können von dort aus weiterverwendet werden.

Dr. Skin schätzt dieses System sehr, da es ihm ermöglicht einfach und effizient von den Vorzügen der Medizininformatik zu profitieren, nebenbei aber sein etwas älteres, lieb gewonnenes Gerät weiterhin benutzen zu können.

6 Use Cases

Die nachfolgenden Use Cases (UC) beschreiben die Abläufe zwischen Benutzer und dem System. Zur Übersicht werden die UC in einer Liste (Use Case List) mit entsprechendem Name aufgeführt, bzw. in einer Map (Use Case Map) grafisch angeordnet.

6.1 Use Case List

Use Case Nr	Use Case Name
1	Bild an DICOM-Server senden (online)
2	Bild an DICOM-Server senden (offline)
3	Synchronisation
4	Storage Commitment
5	Informieren über Synchronisations- und Archivierungszustand

Table 4: Use Case List

6.2 Use Case Map

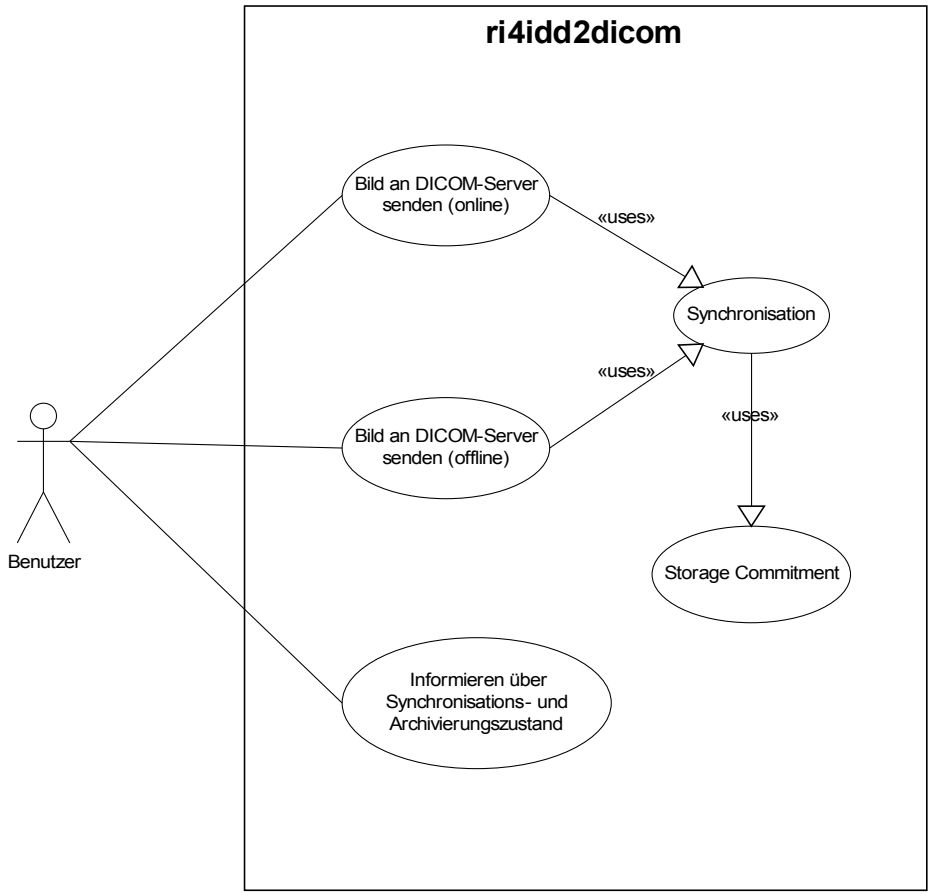


Illustration 1: Use Case Map

6.3 Allgemeine Voraussetzungen/Vorbedingungen

Folgende Vorbedingungen sind für alle Use Cases gegeben, d.h. sie werden nicht mehr explizit in den Vorbedingungen nachfolgender UC aufgeführt:

- Benutzer besitzt ein Diagnosegerät ohne DICOM Unterstützung.
- Benutzer hat den virtuellen Druckertreiber auf dem medizinischen Diagnosegerät installiert, welches ihm ermöglicht Bilder, bzw. PDF Dateien zu generieren.
- Druckertreiber ist entsprechend vorkonfiguriert mit den notwendigen Angaben zum Broker.

6.4 UC1 Bild an DICOM-Server senden (online)

6.4.1 Kurz (brief)

Der Benutzer möchte das soeben erstellte Bild auf seinem medizinischen Diagnosegerät an ein DICOM-Archiv senden und befindet sich im gleichen Netzwerk wie der DICOM-Server.

6.4.2 Vollständig ausformuliert (fully dressed)

Primärer Akteur	Benutzer
Priorität	1
Absicht	Benutzer möchte ein Bild an ein DICOM-Archiv senden.
Vorbedingungen	Benutzer hat eine Bildaufnahme zu einem Untersuch eines Patienten erstellt. Benutzer befindet sich mit medizinischem Diagnosegerät im gleichen Netzwerk wie der DICOM-Server (online-Modus).
Nachbedingungen	Benutzer nimmt Dienst des DICOM-Servers in Anspruch, d.h. das Bild befindet sich danach im DICOM-Archiv.

6.4.3 Erfolgreiches Hauptszenario (main success scenario)

Benutzer Absicht	**System Antwort**
1. Benutzer möchte das Bild, welche er in seinem Bildbetrachtungsprogramm geöffnet hat an ein DICOM-Archiv senden.	
2. Benutzer wählt den entsprechenden virtuellen Drucker aus und startet Druckdialog (klickt auf den Print-Knopf).	
	3. System öffnet Druckdialog für Eingabe der Patienten und die für den Untersuch relevanten Daten (Patient/Studie).
4. Benutzer gibt Daten im Druckdialog ein und startet den „Ausdruck“ (~DICOM-Import).	
	5. System speichert eingegebene Daten in einer Metadatei zusammen mit dem Bild in einem lokalen Cache.
	6. System leitet Daten an DICOM-Broker weiter.

6.5 UC2 Bild an DICOM-Server senden (offline)

6.5.1 Kurz (brief)

Benutzer möchte das soeben erstellte Bild auf seinem medizinischen Diagnosegerät an ein DICOM-Archiv senden; befindet sich nicht im gleichen Netzwerk wie der DICOM-Server.

6.5.2 Vollständig ausformuliert (fully dressed)

Primärer Akteur	Benutzer
Priorität	1
Absicht	Benutzer möchte ein Bild an ein DICOM-Archiv senden.
Vorbedingungen	Benutzer erstellte Bildaufnahme zu einem Untersuch eines Patienten Benutzer befindet sich mit medizinischem Diagnosegerät **nicht** im gleichen Netzwerk wie der DICOM-Server (offline-Modus).
Nachbedingungen	Benutzer nimmt Dienst des DICOM Servers in Anspruch Bild befindet sich auf dem Client in einem lokalen Cache / Warteschlange (Queue) Client wartet mit der Datenübertragung, bis erfolgreiche Verbindung zum Broker sichergestellt wurde.

6.5.3 Erfolgreiches Hauptszenario (main success scenario)

Benutzer Absicht	**System Antwort**
1. Benutzer möchte Bild, welches er in seinem Bildbetrachtungsprogramm geöffnet hat, an ein DICOM-Archiv senden.	
2. Benutzer wählt den entsprechenden virtuellen Drucker aus und startet Druckdialog (klickt auf Print-Knopf).	
	3. System öffnet Druckdialog für Eingabe der Patienten und die für den Untersuch relevanten Daten (Patient/Studie).
4. Benutzer gibt Daten im Druckdialog ein und startet 'Ausdruck' (DICOM-Import)	
	5. System speichert eingegebene Daten in einer Metadatei zusammen mit dem Bild in einem lokalen Cache.
	6. System überprüft Verbindung und erkennt, dass dieses nicht mit im gleichen Netz wie der DICOM-Broker befindet (offline-Modus).
	7. System speichert Daten in einer lokalen Queue den Importauftrag und versucht periodisch die Verbindung mit dem DICOM-Broker herzustellen.

6.6 UC3 Synchronisation

6.6.1 Kurz (brief)

Der DICOM-Client versucht die lokal vorhandenen Daten via DICOM-Broker in ein DICOM-Archiv zu übertragen; dabei besteht eine Verbindung zum DICOM-Broker.

6.6.2 Vollständig ausformuliert (fully dressed)

Primärer Akteur	Client
Priorität	1
Absicht	DICOM-Client versucht die lokal vorhandenen Daten via DICOM-Broker in das DICOM-Archiv zu übertragen.
Vorbedingungen	Unsynchronisierte Daten sind im lokalen Cache des Clients vorhanden. Netzwerkverbindung zu Broker ist vorhanden. Client kennt Konfigurationsparameter für Broker und DICOM-Server/DICOM-Archiv.
Nachbedingungen	Ein Storage Commitment wird erwartet

6.6.3 Erfolgreiches Hauptszenario (main success scenario)

Client Absicht	**System Antwort**
1. prüft ob Verbindung zum Broker hergestellt werden kann.	
2. sendet Daten an Broker.	
	3. Broker prüft ob Verbindung zu DICOM-Server besteht und sendet eine „HELLO“ Nachricht.
	4. Wenn Verbindung o.k., dann antwortet der DICOM-Server auf die „HELLO“ Nachricht.
	5. Broker sendet die vom Client erhaltenen Daten an den DICOM-Server weiter.
	6. DICOM-Server nimmt Daten vom Broker entgegen, prüft diese auf Korrektheit (Vollständigkeit/Konformität) und speichert danach diese Daten im DICOM-Archiv im entsprechendem elektronischen Patientendossier.
	7. Broker wartet auf ein Storage Commitment seitens DICOM-Server.
8. Wartet auf Bestätigung von Broker.	

6.7 UC4 Storage Commitment

6.7.1 Kurz (brief)

DICOM-Broker schickt eine Anfrage an den DICOM-Server, ob die von ihm übergebenen Daten auf dem DICOM-Archiv schon gespeichert worden sind.

6.7.2 Vollständig ausformuliert (fully dressed)

Primärer Akteur	Broker
Priorität	1
Absicht	Broker sendet Anfrage an DICOM-Server, ob Daten bereits im DICOM-Archiv sicher gespeichert worden sind um Nachricht zur Löschung der lokal zwischengespeicherten Daten an den Client weiterzuleiten.
Vorbedingungen	Broker hat eine funktionierende Netzwerkverbindung zum DICOM-Server (online-Modus). Broker hat Auftrag zum Speichern der Daten an den DICOM Server gesendet.
Nachbedingungen	Bild befindet sich im DICOM-Archiv. DICOM-Server sendet Storage Commitment (Speicherbestätigung) an den Broker, dass das Bild erfolgreich im DICOM-Archiv abgelegt wurde. Lokal gespeichert und referenzierte Daten zum Bild sind aus Cache gelöscht.

6.7.3 Erfolgreiches Hauptszenario (main success scenario)

Broker Absicht	**DICOM-Server Antwort**
1. sendet Anfrage an DICOM-Server, ob die von ihm übergebenen Daten im DICOM-Archiv sicher gespeichert worden sind.	
	2. DICOM-Server prüft ob die übergebenen Daten sicher gespeichert worden sind.
	3. DICOM-Server meldet bei erfolgreicher Speicherung dem Broker mit einem Storage Commitment, dass Speicherung funktioniert hat und die Daten sicher im DICOM-Archiv sind.
4. benachrichtigt Client, dass die Daten auf dem DICOM-Archiv sicher gespeichert wurden und somit aus dem lokalen Zwischenspeicher auf dem Client gelöscht werden können.	

6.8 UC5 Informieren über Synchronisations- und Archivierungszustand

6.8.1 Kurz (brief)

Der Benutzer informiert sich im DICOM-Client über den Synchronisations- und Archivierungszustand seiner an den DICOM-Server gesendeten Bilder.

6.8.2 Vollständig ausformuliert (fully dressed)

Primärer Akteur	Benutzer
Priorität	2
Absicht	Benutzer möchte sich über den Synchronisations- und Archivierungszustand informieren.
Vorbedingungen	Benutzer hat bereits via Client ein Bild auf den DICOM-Server gesendet.
Nachbedingungen	Client stellt ein Log über alle Transaktionen dieses Clients zusammen.

6.8.3 Erfolgreiches Hauptszenario (main success scenario)

Benutzer Absicht	**DICOM-Client Antwort**
1. wählt im Client die Log Ansicht aus.	
	2. zeigt dem Benutzer eine Übersicht aller von diesem Client ausgeführten Transaktionen an.
3. informiert sich über die Transaktionen.	

7 Domain Model

Für die zu entwickelnde Applikation wurde folgendes Domain Model ausgearbeitet:

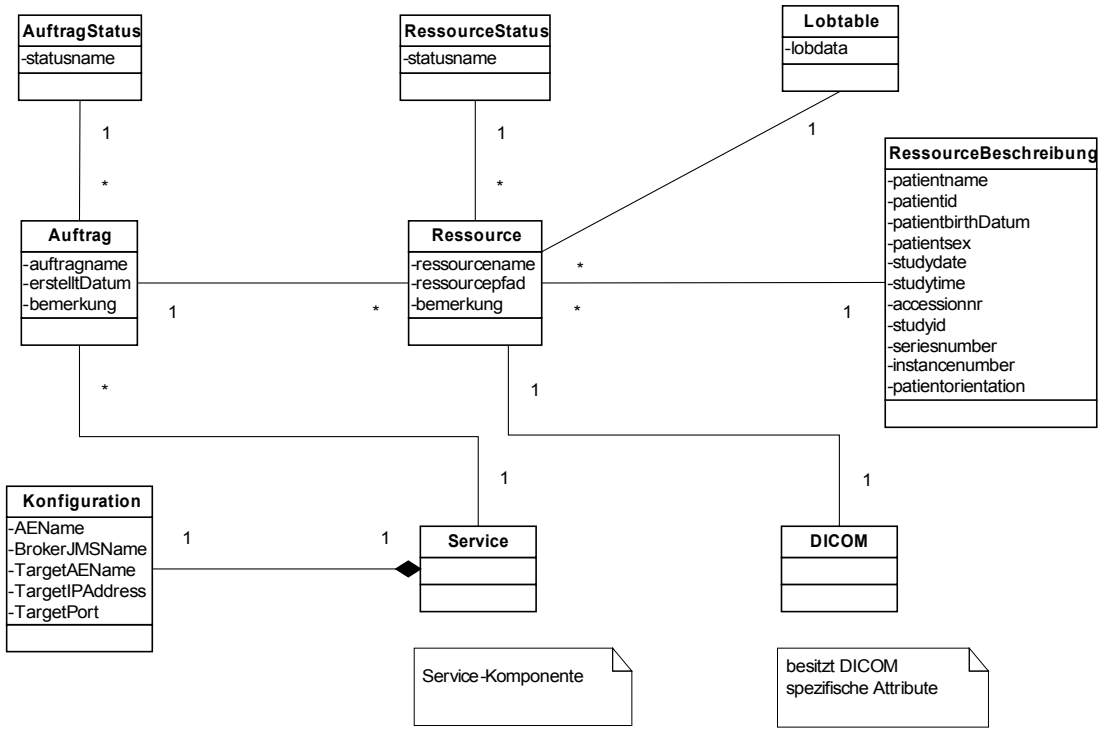


Illustration 2: Domain Model

7.1 Erläuterung

Für das Verwalten der zu importierenden Ressourcen (mit Ressource wäre z. B. eine Jpeg-Bilddatei oder ein PDF-Dokument gemeint) ist ein Verwaltungsobjekt konzipiert worden (Auftrag), welches die entsprechenden Ressourcen (Ressource) mit den spezifischen DICOM-Informationen (DICOM) ergänzt.

Damit der Benutzer, bzw. das System die Möglichkeit hat die zu importierenden Ressourcen bezüglich Importzustand zu unterscheiden, werden die Ressourcen zusammen mit ihrem Auftrag mit entsprechenden Stati versehen (AuftragStatus / RessourceStatus).

Jede Ressource besitzt eine Ressourcenbeschreibung (RessourceBeschreibung), welche die Ressource mit Metadaten (Angaben zu Untersuch und Information des zu behandelnden Patienten wie Name, Vorname, Geburtsdatum, Geschlecht, usw.) identifiziert. Für die Speicherung der ressourcenspezifischen Daten (Bildinformationen, Dokumentinformationen) wurde ein separates Datenbankobjekt vorgesehen (Lobtable), welches diese Daten speichert.

In der zentralen Serviceklasse (Service) wird auf die Konfiguration des Clients zugegriffen und die entsprechenden Aufträge mit den dazugehörenden Ressourcen bearbeitet. Die Konfiguration umfasst dabei die lokalen Konfigurationsvariablen wie z. B. jene für die IP-Adresse, Port des DICOM-Servers, AE Name des Clients und DICOM-Servers, sowie Name der JMS-Queue zur Weiterleitung der DICOM-Daten an den Broker.

8 Aktivitätsdiagramm

Für die zu entwickelnde Applikation wurde folgendes Aktivitätsdiagramm ausgearbeitet, was die entsprechenden Aktivitäten in den nachfolgenden Komponenten (Client- Broker und DICOM-Server) als Blackbox beschreibt:

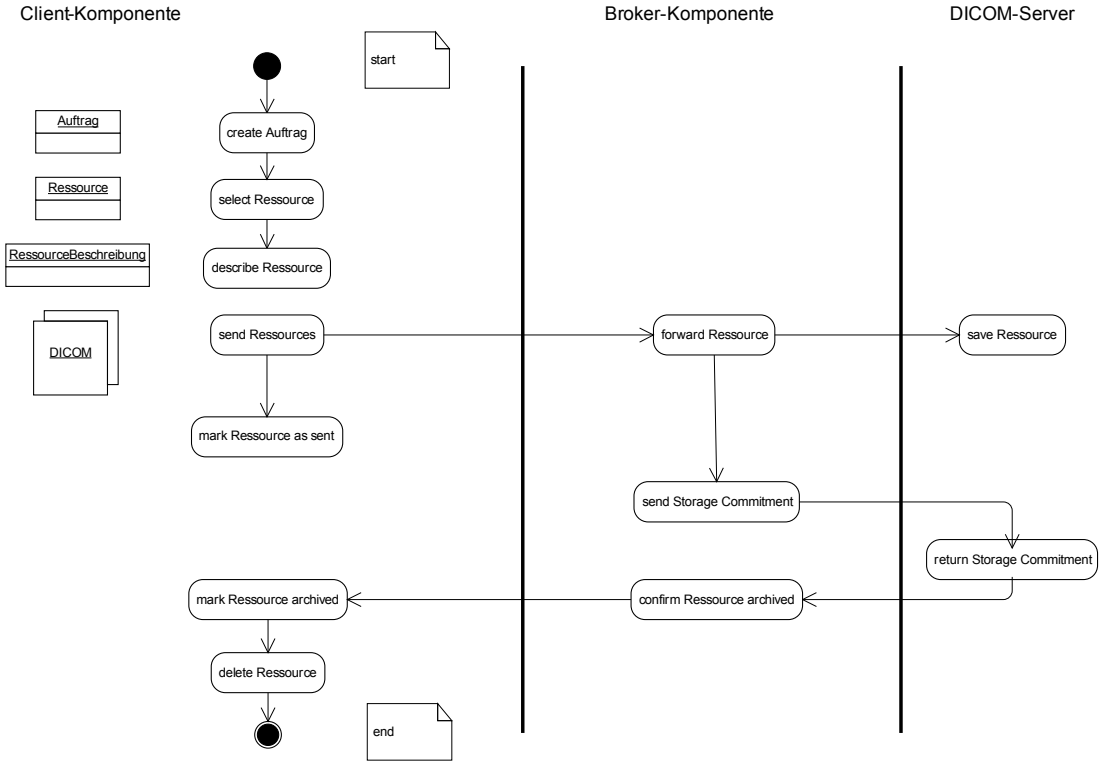


Illustration 3: Aktivitätsdiagramm: Ressource exportieren an DICOM-Server

8.1 Erläuterung

Auf dem Client wird ein neuer Auftrag erstellt. Der Benutzer wird dabei zur Eingabe eines Auftragsnamen aufgefordert, welche diesen beschreibt.

Als nächstes muss der Benutzer für den erstellten Auftrag eine Ressource erfassen. Dieser selektiert eine entsprechende Ressource-Datei (dies kann entweder eine Bild-Datei (*.jpg) oder ein PDF-Dokument (*.pdf) sein) über einen entsprechenden Selektionsdialog. Ist die Ressource selektiert, muss eine entsprechende Beschreibung mit Patientendaten dieser Ressource eingegeben werden.

Wird eine Verbindung zum Broker hergestellt, werden die erstellten Ressourcen in ein DICOM-Objekt gekapselt und an den Broker weitergeleitet. Die weitergeleiteten Ressourcen werden dabei mit dem Status „Sent“ clientseitig vermerkt.

Auf Seite des Brokers wird das DICOM-Objekt entgegengenommen und an den zuständigen DICOM-Server zum Speichern abgegeben.

Der DICOM-Server nimmt das DICOM-Objekt entgegen und speichert dieses in seinem Archiv.

Damit der Client sicherstellen kann, ob die gesendeten Ressourcen im Archiv angekommen ist, muss für die entsprechende Ressource ein Storage Commitment (SC) durch den Broker an den DICOM-Server gesendet werden.

Der DICOM-Server nimmt diese Anfrage entgegen und prüft bei sich im Archiv nach, ob die angegebene Ressource im Archiv sicher archiviert wurde. Ist dies der Fall erhält der Broker eine Bestätigung des Storage Commitment und dieser teilt dem Client dies mit.

Für alle erfolgreichen SC wird die Ressource auf dem Client mit dem Status „archived“ gekennzeichnet.

Alle Ressourcen mit dem Status „archived“ können gelöscht werden, sofern der Auftrag erfolgreich abgeschlossen wurde.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Anforderungsspezifikation

19. Mai 2009

Abstract

Dieses Dokument beinhaltet die Anforderungsspezifikation und Detailspezifikationen zu dieser Studienarbeit.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Anforderungsspezifikation_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
05.03.2009	Dokument erstellt
19.05.2009	Dokument in Version 2 überarbeitet
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument beinhaltet die Anforderungsspezifikation zu dieser Studienarbeit.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Die Spezifikationen sind unterteilt nach Anforderungsspezifikationen und Detailspezifikationen, welche wiederum gegliedert nach Allgemeinen Anforderungen und nach Anforderungen zu den entsprechenden Komponenten (Client, Broker und DICOM-Server) sind.

2 Anforderungsspezifikationen

Die Anforderungen sind so aufgeführt, dass diese testbar sind und für den Anwender von Bedeutung. Die aufgeführten Anforderungen werden mittels eindeutiger Nr. identifiziert um in Testfällen Referenz darauf nehmen zu können.

Nr	**Anforderungsspezifikation**
A1	Ressource wird durch den Benutzer definiert/selektiert und mit Metainformationen (Informationen welche die Ressource beschriebt) versehen
A2	Resssource wird zusammen mit seinen Metainformationen in ein DICOM-Objekt konvertiert
A3	DICOM-Objekt wird an DICOM-Archiv gesendet
A4	Storage Commitment wird verwenden um gesendetes DICOM-Objekt auf dessen sicheren Archivierung zu prüfen
A5	Ressource mit seinen Metainformationen bleiben im System lokal gespeichert, bis Storage Commitment Archivierung bestätigt

Table 1: Anforderungsspezifikationen

3 Detailspezifikationen

Die nachfolgenden Detailspezifikation geben Spezifikationen zur Implementierung der Softwarelösung, welche für den Entwickler relevant sind.

3.1 Allgemeine Anforderungen

- Input
  - Ressource (z. B. eine Jpeg-Bilddatei) muss physikalisch auf dem Dateisystem des Clients vorhanden sein, damit diese zum Import verwendet werden kann
  - Zu jeder Ressource müssen Metadaten vorliegen, welche hauptsächlich eine Beschreibung darstellt, bzw. Informationen zur Untersuchung des Patienten beinhalten und das spätere Wiederfinden auf Seite des Archivs sicherstellt
- Architektur
  - Das System ist Client-Server basiert
  - System gehört in die Gruppe verteilter Systeme, da unterschiedlich Systeme/Komponenten zusammenarbeiten werden
- Komponenten
  - Client wird durch die Funktionalität des Druckertreiber (virtueller Druckertreiber) sichergestellt und auf der Arbeitsstation des Benutzers installiert
  - Eine Brokerkomponente wird als zusätzliches Bindeglied zwischen Client (Druckertreiber) und Server (DICOM-Server) eingeführt um die Importfunktionalität an ein DICOM-Archiv zentral sicherzustellen
  - Der Server ist durch einen DICOM-Server sichergestellt, welcher in der jeweiligen Infrastruktur betrieben wird und als Hauptfunktion ein DICOM-Archiv verwaltet/anbietet
- Datensicherheit
  - Die ins Archiv zu importierenden Daten müssen lokal zwischengespeichert werden um ein Datenverlust zu verhindern, bevor diese nicht im DICOM-Archiv sicher archiviert worden sind
  - Die lokal vorhandene Daten dürfen erst gelöscht werden, wenn diese auf dem DICOM-Server mittels Storage Commitment entsprechend bestätigt wurden.
- Infrastruktur
  - Informationen zur Umgebung muss bekannt sein, z.B. wie heisst das DICOM-Archive (AE Name), wo ist dieser DICOM-Server zu finden (IP-Adressen, Ports des Service) für Kommunikation, bzw. definieren des Zielsystems / Archivs seitens Client

3.2 Client

- Form
  - mittels einem virtuellen Druckertreiber
  - auch als eigenständige Desktop Applikation möglich
- GUI
  - ermöglicht Dialog mit Benutzer
  - anzeigen von Daten und Statusinformationen
- Überprüfung der Benutzereingabe
  - überprüfen/validieren von Datumseingabe (Datumsbereich), Länge der Feldinhalte bei variablen Textfeldern
  - ist die Eingabe falsch, so wird der Benutzer vom System entsprechend benachrichtigt. Dies wird durch eine Anzeige (Pop-Up Fenster oder Statusliste) im entsprechenden Eingabeformular dem Benutzer visualisiert
  - sicherstellen, dass eine gültige Ressource seitens des Benutzers selektiert ist, damit das System diese weiterverarbeiten kann
- Konfiguration
  - Broker-Komponente konfigurierbar
  - Ziel DICOM-Archiv konfigurierbar
  - besitzt einen AE (Application Entity) Namen, welchen die Client Funktionalität (Client-Service) identifiziert
- Datenhaltung
  - speichert lokal die Daten bei sich ab, bis diese gelöscht werden können
- Input
  - Ressource und dazugehörigen Metainformationen
  - Nachricht von Broker, dass entsprechende Ressource gelöscht werden darf
- Output
  - sendet DICOM-Objekt an Broker weiter (behält Daten aber lokal gespeichert)
    - via JMS-Queue

3.3 Broker

- Form
  - als Java Enterprise Applikation
- Funktionalität
  - kennt Clients über ihren AE Namen und IP-Adresse
  - sendet DICOM-Objekt
    - Datenübertragung setzt auf TCP/IP auf
    - verwendet DICOM in der Applikationsschicht
  - sendet Storage Commitment an DICOM-Server um Archivierung des gesendeten Outputs sich bestätigen zu lassen
  - Sicherstellung einer Konfigurationsverwaltung der angeschlossenen Geräte
    - wird für Update des Clients vorgesehen
- Input
  - DICOM-Objekt von Client
- Output
  - sendet DICOM-Objekt an DICOM-Server
  - sendet Storage Commitment an DICOM-Server
    - informiert Client, dass entsprechende Ressource archiviert wurde und gelöscht werden kann
- Sicherheitsaspekte
  - Vertraulichkeit
    - ganze Kommunikation verschlüsselt möglich (Https, SSL)
  - Schutz gegen Manipulationen
    - via Zertifikate/digitale Signaturen
    - Zugriffsberechtigungen
  - Schutz gegen Datenverluste
    - lokaler Datenspeicher als Cache
    - mittels Checksummenfunktion MD5

3.4 DICOM-Server

- DICOM-Server ist eine bestehende Komponente, welche produktiv im jeweiligen medizinischen Umfeld vorgegeben wird
- Speichert/Archiviert DICOM-Objekte in seinem lokalen DICOM-Archiv
- Bietet mittels DICOM Storage Commitment (SCP) einen Service zum Abfragen der von ihm archivierten Ressource an
- Besitzt eine Konfiguration
  - AE Name
  - IP-Adresse / Port
- Besitzt eine Benutzerverwaltung

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Architektur-Design

19. Mai 2009

Abstract

Dieses Dokument beschreibt das Architekturdesign mit den verschiedenen Überlegungen und Architekturentscheide bezüglich der Implementierung dieser Softwarelösung.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Architektur-Design_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
30.03.2009	Dokument erstellt
19.05.2009	Dokument in Version 2.0 überarbeitet
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument beschreibt das Architekturdesign mit den verschiedenen Überlegungen und Architekturentscheide bezüglich der Implementierung dieser Softwarelösung.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Das Dokument ist unterteilt in die Hauptbereiche Architektonische Darstellung, Architektonische Ziele und Einschränkungen, Funktionalität, Logische Architektur, Kommunikation, Prozesse und Threads sowie Systemeigenschaften.

2 Architektonische Darstellung

In diesem Abschnitt werden die Konzepte beschrieben, welche sich über die ganze Softwarelösung hinweg ziehen. Es wurden zwei Szenarien in der Design-Phase ausgearbeitet, die folgend erläutert werden.

2.1 Szenario 1 (ohne Broker-Komponente)

2.1.1 Big Picture

Das folgende UML Deployment-Diagramm veranschaulicht die unterschiedlichen Systeme-/Systemkomponenten, in welchem die entwickelte Softwarelösung betrieben wird.

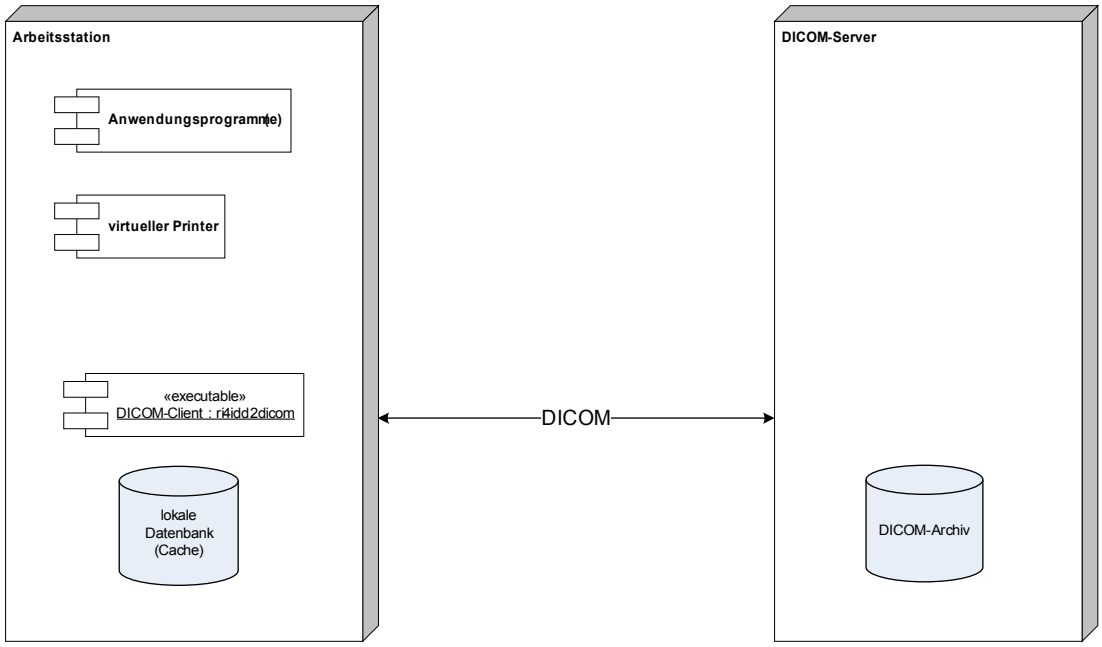


Illustration 1: Big Picture (UML Deployment-Diagramm) Szenario 1

Die Softwarelösung besteht aus einem Client (ri4idd2dicom-Client), der auf einer Arbeitsstation des Anwenders installiert wird (siehe Abbildung oben links).

Auf der Gegenseite (siehe Abbildung oben rechts) arbeitet ein DICOM-Server, welcher in der Infrastruktur bereits vorgegeben und für das Speichern/Archivieren der elektronischen Patientendaten zuständig ist.

Der Client (ri4idd2dicom-Client) kann auf mehreren Arbeitsstationen gleichzeitig betrieben werden. Diese kennen sich untereinander aber nicht, arbeiten daher autonom. Kommuniziert wird direkt über ein DICOM-Applikationsprotokoll.

2.1.2 Komponentenübersicht

Die Softwarelösung ri4idd2dicom wurde als Client Lösung konzipiert. Jede einzelne Komponente wird nachfolgend beschrieben. Inhalt dabei sind ihre spezifischen Eigenschaften, die Kommunikation unter den Komponenten, sowie die Art und Weise, wie diese Komponente implementiert werden müssen.

2.1.2.1 Client-Komponente

Der Client (ri4idd2dicom-Client) wird lokal auf der Arbeitsstation des Anwenders installiert und stellt dem Benutzer eine Schnittstellen-Funktionalität auf eine entsprechendes DICOM-Archiv zur Verfügung. Der Client ist Java basiert und enthält ein einfaches User-Interface (UI) für die Interaktion mit dem Benutzer.

In einer lokal bei sich geführten Datenbank speichert der Client temporär alle zu importierenden Ressourcen (z.B. *.jpg-Bilddateien) mit entsprechenden Angaben zum Patient und durchgeführten Studie (Metadaten) ab.

Mittels Client-seitigen Voreinstellungen, welche über ein Optionsmenü verändert werden können, bietet der Client ein Integrationsmöglichkeiten auf andere DICOM-Archive innerhalb der gleichen Systemumgebung (Netzwerk) an.

Der Client ist verantwortlich zum Erstellen der DICOM-Dateien, welche er mittels den lokalen Ressourcen inkl. Metadaten entsprechend umwandeln kann.

Folgende Informationen müssen seitens Client bekannt sein:

- AE (=Application Entity)
- Zielarchiv (Option sagt aus, in welches Archiv Client die Daten speichern muss)

Für die Kommunikation wird ein DICOM-spezifisches Applikationsprotokoll verwendet, welches auf TCP-IP aufsetzt.

Zusammengefasst kann die Client-Komponente als Einzelbenutzer Swing-JDBC-Applikation beschrieben werden, welche lokal eine HSQLDB als Datenspeicher einsetzt und mittels DICOM Applikationsprotokoll direkt mit einem DICOM-Server kommuniziert.

2.1.2.2 Server-Komponente

Die Server-Komponente ist durch den DICOM-Server vorgegeben, welcher für den Betrieb dieser Softwarelösung zwingend voraus zusetzen ist. Dieser Server sollte in der heutigen Zeit in allen gängigen Bereiche der Medizin schon produktiv im Einsatz stehen und wertvolle Dienste in der täglichen Arbeit leisten, ein daher unverzichtbares Arbeitsinstrument. Ein DICOM-Server könnte beispielsweise jener aus der dcm4che.org Open Source Lösung sein, welcher ein DICOM-Archiv anbietet.

Der DICOM-Server betreibt bei sich lokal eine eigene Datenbank (DICOM-Archiv). Diese Datenbank könnte beispielsweise eine PostgreSQL-DB sein, oder auch andere verbreitete Relationale Datenbanken Systeme. In der Datenbank werden alle Patientendaten, Bilder usw. zentral abgespeichert.

Folgende Eigenschaften muss der DICOM-Server haben:

- Besitzt eine Netzwerkverbindung an sein Klinik-Netzwerk via Netzwerkkarte
- Netzwerkkarte besitzt gültige IP-Adresse
- Port, auf welchem die DICOM-Services zu Verfügung stehen
- DICOM AET (= Application Entity Title) des Servers ist bekannt, z. B. „PACSSERVER“, welche die Applikation benennt

2.2 Szenario 2 (mit Broker-Komponente)

2.2.1 Big Picture

Das folgende UML Deployment-Diagramm veranschaulicht die unterschiedlichen Systeme-/Systemkomponenten, in welchem die entwickelte Softwarelösung betrieben wird.

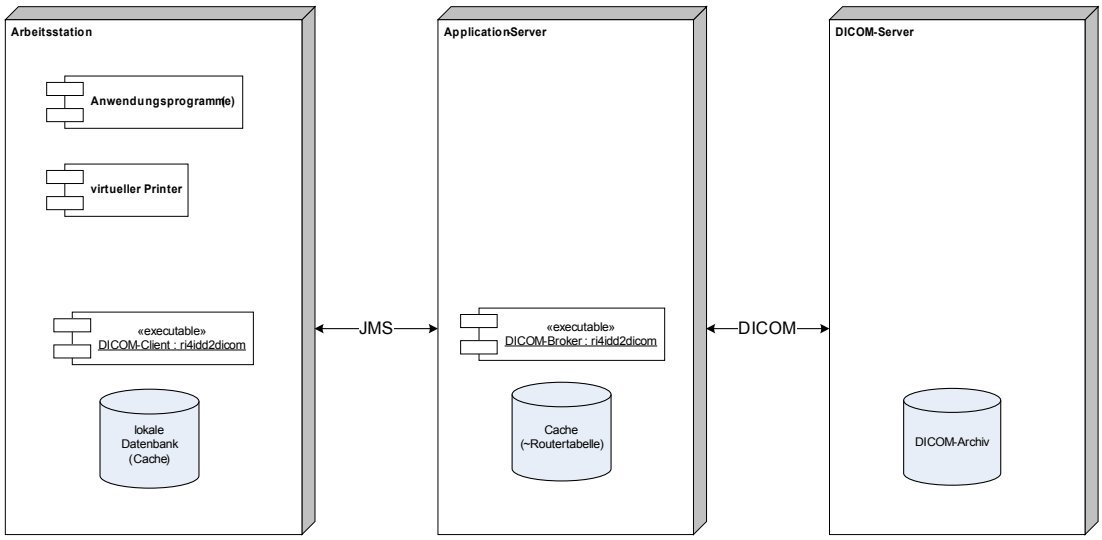


Illustration 2: Big Picture (UML Deployment-Diagramm) Szenario 2

Dieses Diagramm ist gleich wie jenes aus dem Szenario 1, besitzt aber zusätzlich eine Broker-Komponente (ri4idd2dicom-Broker), welcher als Vermittler zwischen Arbeitsstation und DICOM-Server arbeitet (siehe Abbildung oben Mitte).

Der Broker (ri4idd2dicom-Broker) wird innerhalb eines Application-Servers betrieben, welcher nur einmal vorhanden sein muss.

Der Client (ri4idd2dicom-Client) kommuniziert mit dem Broker über eine JMS-Queue, welche nachrichtenbasiert und asynchron ist. Der Broker nimmt die Nachrichten von den Clients entgegen und leitet seinerseits die darin gespeicherten DICOM-Objekte an den jeweiligen DICOM-Server mittels DICOM-Applikationsprotokoll weiter.

Um sicherzustellen, ob die DICOM-Objekte auch sicher auf dem DICOM-Server archiviert wurden, muss der Broker für jedes gesendete DICOM-Objekt ein Storage Commitment ausführen.

2.2.2 Komponentenübersicht

Die Softwarelösung ri4idd2dicom wurde als Client-Server Lösung konzipiert. Jede einzelne Komponente wird nachfolgend beschrieben. Inhalt dabei sind ihre spezifischen Eigenschaften, die Kommunikation unter den Komponenten, sowie die Art und Weise, wie diese Komponente implementiert werden müssen.

2.2.2.1 Client-Komponente

Die Client-Komponente ist gleich der aus dem Szenario 1 (siehe 2.1.2.1), besitzt zusätzlich aber noch eine Konfiguration des Brokers (ri4idd2dicom-Broker).

Zudem sendet der Client die DICOM-Objekte nicht mehr direkt an den DICOM-Server, sonder an den in der Konfiguration vorgesehenen Broker.

Folgende Informationen müssen seitens Client zusätzlich bekannt sein:

- JMS-Queue (Angabe des JMS-Queue Namen, in welchem der Client die DICOM-Objekte senden - wird verwendet zur Kommunikation mit ri4idd2dicom-Broker)

2.2.2.2 Broker-Komponente

Die Funktionalität des Servers in dieser Softwarelösung wird sichergestellt durch eine Broker-Komponente (ri4idd2dicom-Broker), welche mehrere Clients gleichzeitig bedienen kann. Der Broker leitet die erhaltenen DICOM-Objekte an einen DICOM-Server und ist zudem verantwortlich, dass für jedes DICOM-Objekt ein Storage Commitment ausgeführt wird.

Der Broker besitzt dabei einen lokalen Cache, vergleichbar mit einer „Router Tabelle“. Diese speichert einerseits die Clients mit ihren AE Namen, sowie die gesendeten DICOM-Objekte und einem zusätzlichen Flag um das Storage Commitment zu speichern. Wurde das Storage Commitment durch den DICOM-Server bestätigt, wird der Client vom Broker benachrichtigt, dass die entsprechende Ressource lokal gelöscht werden kann und sein lokaler Cache geleert.

Durch die Indirektion über einen Broker, welcher losgelöst vom DICOM-Server betrieben wird, erzielt man eine grosse Flexibilität und wenig Abhängigkeiten für den Import von Daten an ein DICOM-Archiv. Der Broker ist die zentrale Komponente und zuständig für den DICOM-Datenimport zwischen Client (ri4idd2dicom-Client) und DICOM-Server und die Logik zum DICOM-Datenimport muss nur an einer Stelle unterhalten werden.

Mit dieser gewonnenen Flexibilität könnte man zusätzlich Möglichkeiten für ein zentrales Logging, bzw. Management der Clients vorsehen. Dies ist nur eine Idee, wurden in dieser Arbeit aber nicht weiter konkretisiert.

2.2.2.3 Server-Komponente

Für die Beschreibung der Server-Komponente wird auf den Abschnitt (2.1.2.2 Server-Komponente) verwiesen, da diese sich gleich bleibt.

Der Einzige Unterschiede besteht darin, dass der Broker den DICOM-Import macht, anstelle des Clients selbst.

2.2.3 Erweiterte Lösung

Um den DICOM-Import noch eine Stufe weiter abstrahieren zu können wäre denkbar, dass die DICOM-Funktionalität vom Client ganz entfernt und auf den Broker ausgelagert wird.

Mit dieser Lösung würden die DICOM spezifischen Arbeiten nur an einer Stelle durchgeführt, was von Vorteil ist, da der Broker sich immer im gleichen Netzwerk wie der DICOM-Server befindet. Zudem wäre der DICOM-Server für die Clients transparent, da die Clients ihre Daten immer an den Broker senden und nichts vom DICOM-Server / über DICOM wissen müssen.

Das Zwischenspeichern der Ressourcen muss aber nach wie vor auf dem Client stattfinden, da dieser ja auch offline, d.h. nicht im gleichen Netzwerk wie der DICOM-Broker. betrieben werden muss.

3 Architektonische Ziele und Einschränkungen

Dieser Abschnitt beschreibt die Ziele und Einschränkungen, welche Einfluss auf die Architektur haben.

3.1 Safety

Um die Sicherheit der Daten zu gewährleisten werden verschiedene Techniken innerhalb des Softwareprojekts eingesetzt, welche nachfolgend kurz erklärt werden:

3.1.1 DICOM Konformität

Um eine reibungslose Integration zu anderen DICOM-Systemen gewährleisten zu können, müssen die erstellten DICOM-Daten eine Konformität zum DICOM-Standard aufweisen. DICOM Konformität wird als Begriff verwendet, welcher aussagt, dass diese DICOM-Daten gemäss dem DICOM-Standard richtig erstellt wurden.

Für eine Applikation heisst dies konkret, dass die Benutzereingaben überprüft werden müssen und spätestens beim Umwandeln von Daten in eine DICOM-Datei die richtigen, gültigen Felder/Attribute gesetzt werden müssen.

Wenn DICOM Konformität gewährleistet ist, dann gehen keine Daten verloren, bzw. werden nicht vom DICOM-Server zurückgewiesen.

3.1.2 DICOM Storage Commitment

Mittels der DICOM Storage Commitment Funktionalität kann bei einer automatischen Archivierung die erfolgreiche Speicherung auf dem Archiv durch den DICOM-Server bestätigt werden.

Wird die Storage Commitment Meldung des Clients durch den DICOM-Servers erfolgreich quittierten, kann die lokal vorhandene Ressource gelöscht werden.

Die Verantwortlichkeit zum Aufbewahren der Ressource (DICOM-Objekt) wurde dem DICOM-Archiv übertragen.

3.2 Security

Da die Softwarelösung keine Einschränkungen seitens der Anwendung durch die Benutzer vorsieht, existiert keine Benutzerverwaltung. Es muss lediglich sichergestellt werden, dass die Daten nicht irrtümlicherweise modifiziert werden können.

Da die selektiere Ressource lokal aus dem Dateisystem gelöscht, oder modifiziert werden könnte, wurde seitens der Softwarelösung eine temporäre Speichermöglichkeit der Ressourcen vorgesehen, welche für den Benutzer nicht sichtbar ist.

Dies wird mit dem Einsatz einer lokalen Datenbank sichergestellt. Die DB bietet nebst dem Speichern der Meta-Informationen (Patienten Daten) auch die Möglichkeit die Ressource in einer LOB Tabelle zu speichern.

Damit die Daten nicht modifiziert werden können, muss die Datenbank vor Fremdzugriff geschützt sein. Dies kann mittels einer Administrations-UserID und dem dazugehörigen Passwort sichergestellt werden.

Seitens DICOM-Server gibt es Benutzer mit unterschiedlichen Rollen. Für den Import wird

ein speziell definierter Benutzer verwendet, welcher auf dem System eingerichtet werden muss. Für die Datenabfrage können unterschiedliche Benutzer zusätzlich eingerichtet werden, welche unterschiedliche Rechte (Lösch-, Editier- oder nur Leserecht) besitzen.

3.3 Privacy

Die Kommunikation der Komponenten Client / Broker / DICOM-Server findet in einem privaten Netzwerk statt, welches entsprechend geschützt sein sollte. Daher werden die Daten vom Client auf den Broker, bzw. die Daten vom Broker an den DICOM-Server unverschlüsselt übertragen.

Seitens der Open Source DICOM Implementierung dcm4che, wäre eine verschlüsselte Kommunikation möglich, dies muss aber speziell eingerichtet, bzw. in der Implementierung der Softwarelösung entsprechend programmiert werden.

Im Rahmen dieser Studienarbeit wird die verschlüsselte Kommunikation nicht eingesetzt.

4 Funktionalität

Gegeben durch die Funktionalen Anforderungen sind folgende drei Hauptbereiche bezüglich Funktionalität identifiziert worden:

- Client-Funktionalität (mit GUI)
- Broker-Funktionalität (ohne GUI)
- DICOM spezifische Funktionalität

4.1 Systemsequenzdiagramm

Nachfolgendes Systemsequenzdiagramm zeigt den Ablauf eines DICOM-Importes mittels der zu entwickelnden Softwarekomponente (ri4idd2dicom Client/Broker) auf.

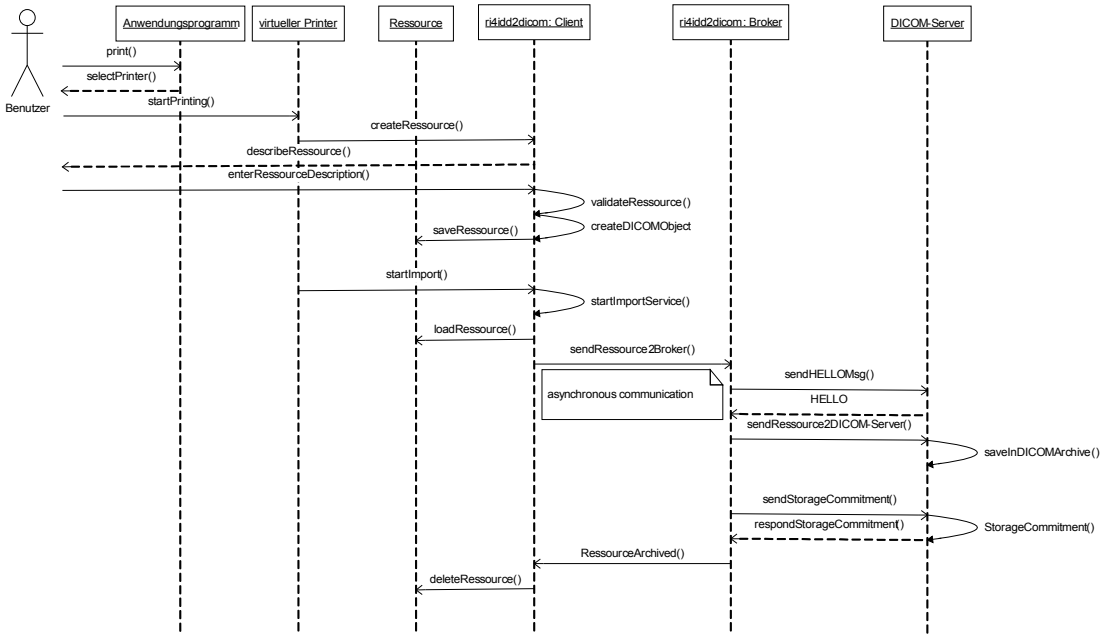


Illustration 3: Systemsequenzdiagramm DICOM-Import

4.1.1 Erläuterung

Der Benutzer (ganz links in der obigen Grafik) hat ein Anwendungsprogramm geöffnet, z.B. ein Bildbetrachter. Mittels Druckfunktion möchte er das aktuell im Anwendungsprogramm angezeigte Bild in ein DICOM-Archiv importieren. Dazu wählt er den passenden virtuellen Printer aus, welche die DICOM-Archivierungs-Importfunktion zu Verfügung stellt.

Der virtuelle Printer erzeugt via dem Client im Hintergrund eine Ressource, welche das Bild enthält. Als nächstes muss die Ressource mit Metadaten (Angaben zu Untersuch und Patienteninformationen) versehen werden. Der Benutzer wird aufgefordert diese einzugeben. Liegen diese Informationen clientseitig vor, werden diese auf Vollständigkeit und Korrektheit überprüft und falls o.k. in ein DICOM-Objekt umgewandelt und abgespeichert.

Ist die Importfunktion durch den virtuellen Druckertreiber ausgelöst worden (kann benutzer- oder zeitgesteuert sein), läuft auf dem Client der Import-Service. Dieser lädt alle für den Import bereitstehenden Ressourcen (DICOM-Objekte) aus seinem Cache und sendet diese einzeln an einen zentralen Broker weiter.

Der Broker erhält vom Client ein DICOM-Objekt, welches er an ein DICOM-Server weiterleiten muss. Damit die Kommunikation sichergestellt ist, sendet der Broker eine Kontrollnachricht (HelloMsg) um die Funktionalität des DICOM-Archives zu überprüfen. Ist diese erfolgreich, so leitet er das DICOM-Objekt entsprechend weiter. Der DICOM-Server nimmt das DICOM-Objekt entgegen und speichert dieses in seinem Archiv ab.

Damit die Ressourcen lokal auf dem Client wieder freigegeben werden können, muss der Client sicher sein, dass die Ressource auf dem DICOM-Server in dessen Archiv korrekt archiviert wurde. Dazu sendet der Broker für die entsprechende Referenz (Instance-UID) der Ressource ein Storage Commitment an den DICOM-Server. Der DICOM-Server überprüft die Speicherung dieser Ressource und sendet bei erfolgreicher Archivierung seinerseits dem Broker eine positive Antwort zurück. Der Broker teilt dies dem Client mit, damit dieser die Ressource lokal bei sich aus dem Cache löschen kann.

4.2 Zustandsdefinitionen

Damit die Applikation dem Benutzer einen Überblick des Auftrags und dessen zu importierenden Ressourcen wiedergeben kann, werden die Aufträge und die dazugehörigen Ressourcen mit einem Status versehen (Auftragsstatus, bzw. Ressourcestatus).

Die Stati werden anhand den definierten Übergangsbedingungen durch die Applikation vergeben und in der jeweiligen Ansicht im GUI angezeigt.

Folgende Stati sind innerhalb der Applikation vorhanden:

4.2.1 Auftragsstatus

Status	**Beschreibung**
New	Ein neuer Auftrag wurde angelegt
Pending	Abarbeitung des Auftrags ist pendent, d.h. es wurden noch nicht alle Ressourcen archiviert
Done	Auftrag wurde erfolgreich abgearbeitet, d.h.alle Ressourcen welche in diesem Auftrag enthalten sind, wurden erfolgreich archiviert
Error	Innerhalb eines Auftrages existiert eine Ressource mit einem Fehler

Table 1: Beschreibung für Status zum Auftrag

4.2.2 Ressourcestatus

Status	**Beschreibung**
Created	Eine neue Ressource wurde erstellt
Sent	Die Ressource wurde an den Broker versendet
Archived	Die Ressource wurde im DICOM-Archiv archiviert (Storage Commitment war erfolgreich)
Error	Es gab ein Fehler (z.B. Umwandlung in ein DICOM-Objekt, Ressource fehlerhaft, u.s.w.)

Table 2: Beschreibung für Status zur Ressource

4.3 Zustandsübergänge

Die im Zustandsdiagramm dargestellten Zustände und Übergänge visualisieren die in der Applikation vordefinierten Zustandsübergänge für einen Auftrag mit entsprechenden Ressourcen.

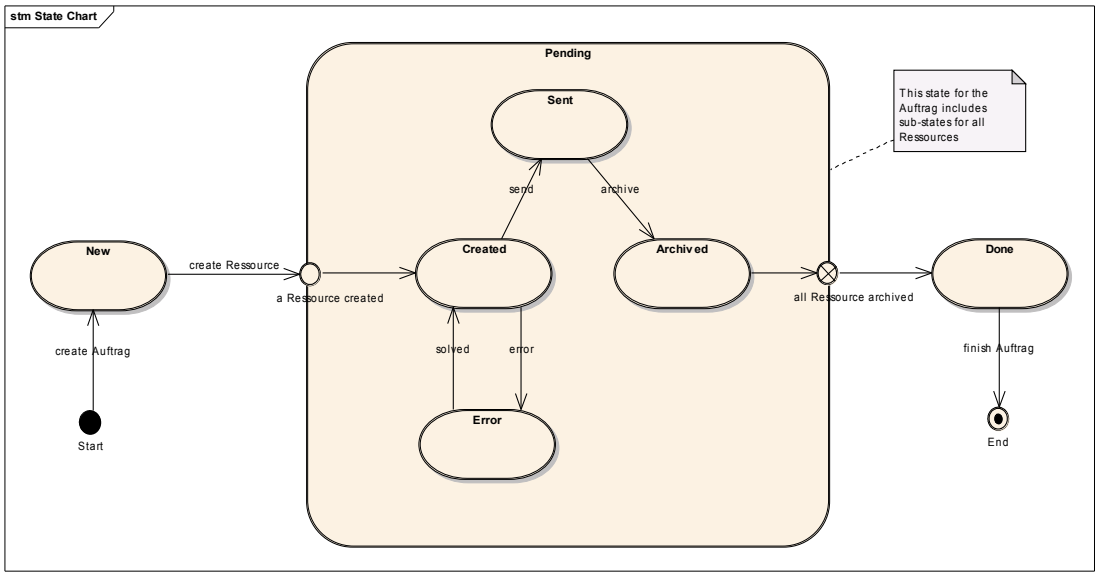


Illustration 4: Zustandsdiagramm Client

4.4 DICOM-Sendekomponente

Die DICOM-Sendekomponente stellt die Funktionalität zum Senden der lokal auf dem Client (ri4idd2dicom-client) zwischengespeicherten Daten über einen Broker (ri4idd2dicom-Broker) an ein DICOM-Archiv sicher.

Ist die Verbindung im Online-Modus, so werden alle für den Import vorgesehenen Ressourcen auf dem Client in ein passendes DICOM-Objekt umgewandelt und an den Broker gesendet. Ist die Ressource versendet, so wird diese mit dem Ressourcestatus „Sent“ gekennzeichnet.

Schlägt die Verbindung zum DICOM-Broker fehl, so wird der Benutzer mittels einer Nachricht informiert und anschliessend wieder in den Offline-Modus gewechselt.

4.5 Client Verbindungsmodi

Der Benutzer hat die Möglichkeit via GUI eine Verbindung mit dem Broker herzustellen. Je nach Verbindungsstatus wird in die folgenden Modi gewechselt:

Modus	**Beschreibung**
Online-Modus	Client ist im gleichen Netzwerk wie der Broker, d.h. alles was im lokalen Cache ist muss an die JMS Queue gesendet werden. Die DICOM-Sendekomponente wird gestartet.
Offline-Modus	Client ist nicht im gleichen Netzwerk wie der Broker, d.h. es besteht keine Verbindung – die DICOM-Objekte müssen lokal zwischengespeichert/gecached werden.

Table 3: Client Verbindungsmodi

5 Logische Architektur

5.1 Übersicht

Für jede Komponente (Client, Broker) sind eigenständige Architekturdesigns vorgenommen worden.

5.2 Client

5.2.1 Architektur

Der Client wurde nach einer 3-Tier Architektur gegliedert. Die Architektur beinhaltet ein Graphical User Interface (GUI) Layer, einem Domain Layer und einem Technical Services Layer. Die vorgenommene Fachliche Unterteilung (vertikal) gliedert die Architektur und definiert entsprechende Verantwortlichkeiten. Es wird jeweils immer von den oberen Schichten auf die unteren zugegriffen, nicht umgekehrt.

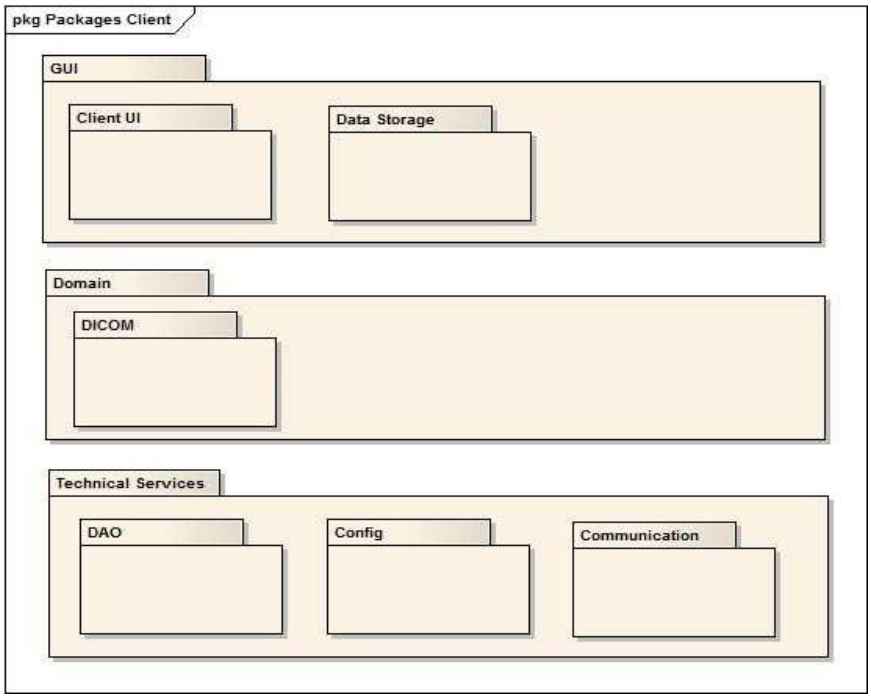


Illustration 5: Architektur Client

5.2.1.1 GUI

Im GUI Layer befinden sich alle Designelemente, welche zum Anzeigen der User Interface Komponenten nötig sind. Des weiteren befindet sich hier noch den Bereich der Datenspeicherung (Persistenzframework).

5.2.1.2 Domain

Im Domain Layer befinden sich alle Komponenten, welche die Hauptfunktionalität der Applikation sicherstellen. Es sind dies DICOM spezifischen Funktionalitäten, z. B. Umwandlung der Ressource in ein DICOM-Objekt.

5.2.1.3 Technical Services

Im Layer Technical Services sind alle Komponenten angesiedelt, welche für die Applikation spezielle Services zu Verfügung stellen. Es sind dies einerseits die DAO Komponenten, die Configuration sowie die Communication Komponente.

5.2.2 Design-Pakete

5.2.2.1 Package GUI

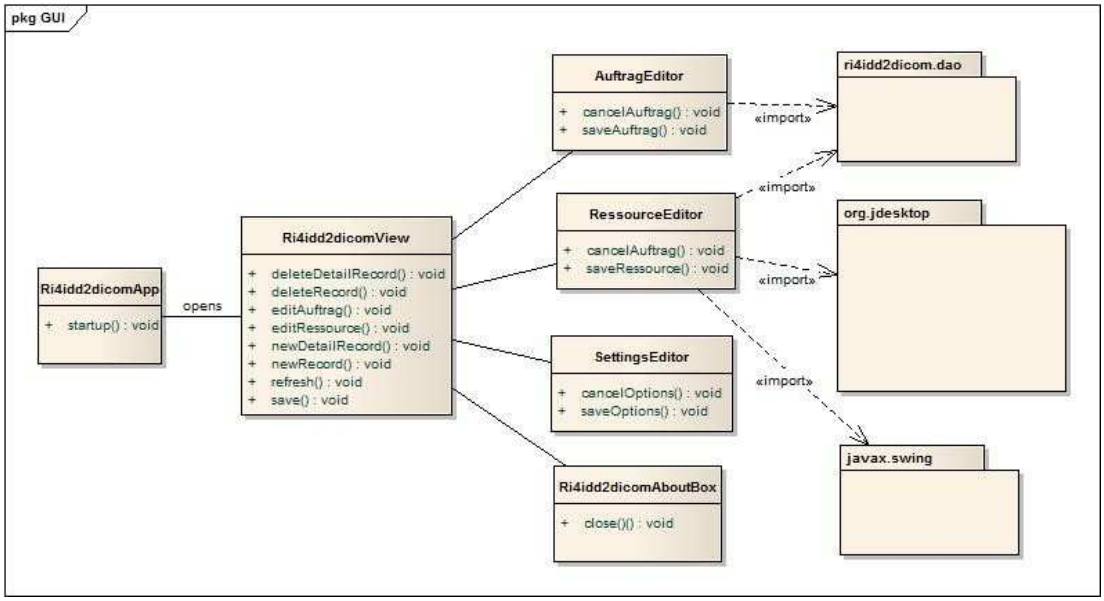


Illustration 6: Package GUI

Ich habe mich für den Einsatz des Swing Application Framework[1] (JSR-296) entschieden, welches eine Open Source Implementierung ist. Das Swing Application Framework ist eine Prototyp Implementation, basierend auf einer kleinen Menge Java Klassen, welches das Erstellen einfache Desktop Applikationen vereinfacht (siehe Package org.jdesktop). Dieses Framework ist noch in Entwicklung, diese JSR sollte aber in der nächsten Java Version 7 integriert sein.

Nachfolgende Komponenten sind hauptverantwortlich für das GUI.

Komponentenname	**Beschreibung**
AuftragEditor.java	Erfassungsform für Aufträge
RessourceEditor.java	Erfassungsform für Ressourcen
Ri4idd2dicomAboutBox.java	Form, zeigt Infos zu der Applikation an
Ri4idd2dicomApp.java	Basisklasse dieser Applikation, startet die Applikation
Ri4idd2dicomView.java	Hauptfenster (Menübar, Ansichten, Statusbar)
SettingsEditor.java	Erfassungsform für Client Einstellungen

Table 4: Komponenten Package GUI

1 http://appframework.dev.java.net/

Die GUI-Komponenten arbeiten nach dem Beans Binding Standard (JSR 295)[2], welcher erlaubt zwei Eigenschaften (Typischerweise zwei Objekte) in Synchronisation zu halten. Der Schwerpunkt liegt dabei an der Möglichkeit Swing Komponenten (Bsp. JTable mit ihrem Model) zu binden.

5.2.2.2 Package Domain

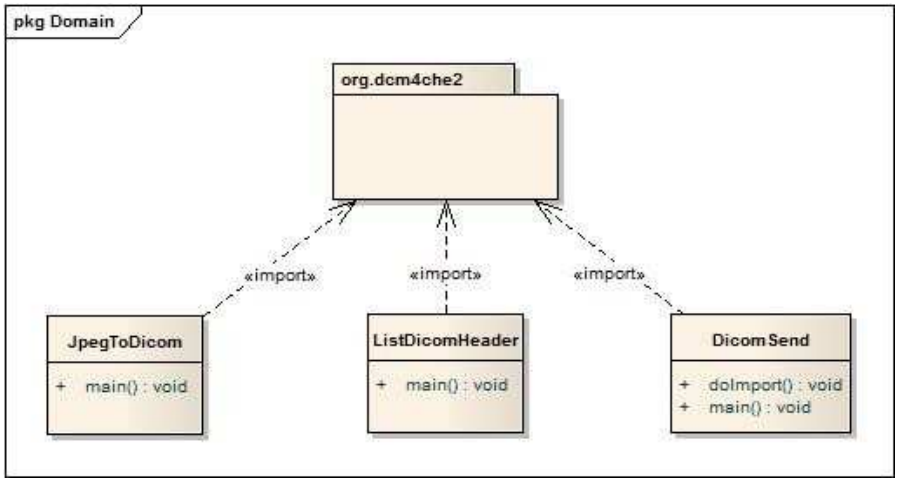


Illustration 7: Package Domain

Diese Klassen sind für die DICOM spezifischen Funktionalitäten verantwortlich. Für Operationen mit DICOM wird dabei auf das dcm4che2 Toolkits zurückgegriffen, welche die Funktionalität in notwendigen Jar-Librarys zur Verfügung stellt.

Komponentenname	**Beschreibung**
DicomSend.java	Zum direkten Senden der DICOM-Objekte an den DICOM-Server (Broker wird ausgelassen)
JpegToDicom.java	Konvertiert eine Jpeg Datei in ein DICOM-Objekt
ListDicomHeader.java	Listet den DICOM-Header eines DICOM-Objektes auf

Table 5: Komponenten Package Domain

5.2.2.3 Package Technical Services:

In diesem Package befinden sind die DAO Klassen (alles Entity Beans), sowie die Klasse zum senden der DICOM-Objekte an den Broker

Komponentenname	**Beschreibung**
Auftrag.java	Entity Bean für einen Auftrag
AuftragStatus.java	Entity Bean für einen Auftragstatus
Lobtable.java	Entity Bean für ein LOB
Ressource.java	Entity Bean für eine Ressource
RessourceBeschreibung.java	Entity Bean für eine Ressourcen Beschreibung
RessourceStatus.java	Entity Bean für einen Ressourcestatus

Table 6: Komponenten Package Technical Services

2 https://beansbinding.dev.java.net/

5.2.3 Datenspeicherung

Für die Datenspeicherung seitens des Clients wird lokale eine Datenbank eingesetzt. Mittels dieser Datenbank kann einerseits die Problematik der Objekt-Serialisierung innerhalb der Applikation, andererseits das Caching der Ressource-Dateien (*.jpg Dateien) gelöst werden.

Ich habe mich für eine HSQLDB[3] (HyperSQL DataBase) als Datenbank entschieden, da diese eine vollständig in Java programmierte relationale SQL-Datenbank ist. HSQLDB zeichnet sich durch ihre kleine Grösse aus und die Funktionalität kann sehr einfach in ein Projekt mittels Einbindung einer einzigen JAR-Datei (hsqldb.jar) realisiert werden. Es wird zudem ein JDBC-Treiber zur Verfügung gestellt und die Tabellen werden innerhalb Dateien auf der Festplatte des Clients abgespeichert. Die Datenbank wird im Embedded Modus betrieben, d.h. Wird dieses Beim Starten und Beenden des Clients entsprechend geladen und heruntergefahren.

Mittels JPA[4] (Java Persistent API) werden die Daten aus der Datenbank mit jenen des Domain-Models (DAO Klassen) verknüpft. Werden Daten verändert, so werden diese über ein Transaktion im System gekennzeichnet. Wird vom Benutzer ein Speichern der Daten ausgelöst, dann wird die Transaktion mittels Commit gespeichert, bzw. bei einem Fehler oder Abbrechen des Speicherbefehls mittels einem Rollback zurückgesetzt. Diese Technik erlaubt einen konsistenten Datenzustand auf der Datenbank.

Für die Speicherung der Systemeinstellungen wird ein lokales Properties-File verwendet.

5.2.3.1 Relationales Datenbankmodell

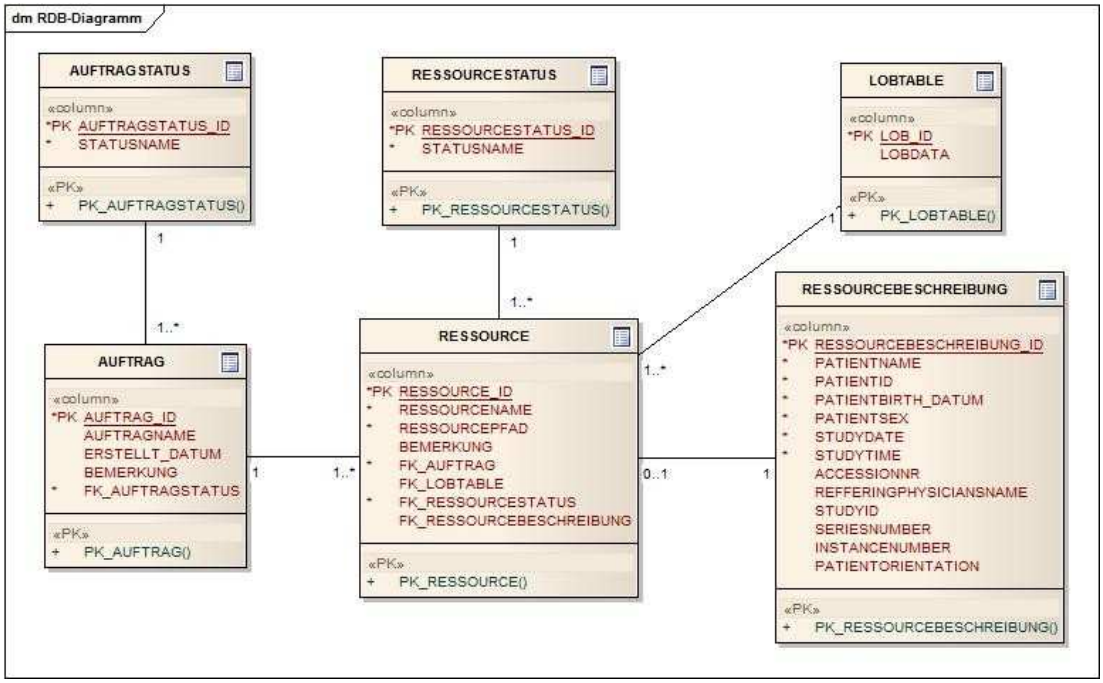


Illustration 8: Relationales Datenbankmodell Client

Das Relationale Datenbankmodell beschreibt die in der Datenbank erstellten Tabellen. Jede Tabelle ist mit einem entsprechenden Primary-Key (Primärschlüssel) versehen. Via Foreign-Key (Fremdschlüssel) wird eine Beziehung zu den Daten hergestellt.

3 http://hsqldb.org/
4 http://java.sun.com/javaee/technologies/persistence.jsp

5.2.3.2 Abfragen

Für die Datenbankabfragen wird auf die Java Persistence Query Language aufgesetzt, welche einem erlaubt mit einer einfachen Abfrage direkt auf die darunter liegenden Datenspeicherung zuzugreifen.

5.2.3.3 Anmerkung zur Datenspeicherung bezüglich Sicherheit

Bei Starten der Applikation wird jedes mal die Datenbank geladen. Dabei wird ein Zugriff unter Angabe eines Datenbank-Benutzers mit entsprechendem Passwort hergestellt. Bei einem produktiven Einsatz der Applikation ist ein Passwort zu wählen, dass eine gute Komplexität aufweist und den Zugriff auf die Datenbank schützt.

Gegeben durch die Architektur der HSQLDB ist ein Nachteil klar gegeben. Dieser zeigt sich beim abspeichern der Daten nach dem Beenden der Applikation, da diese innerhalb des Filesystems in Dateien abgelegt werden, welche ev. durch den Benutzer einsehbar ist. Dies könnte durch Einschränken des Lesezugriffs, bzw. dem verschlüsselten Ablegen der Daten innerhalb der Datenbank gelöst werden.

5.3 Broker

5.3.1 Architektur

Befindet sich als eine Komponente auf einem Application-Server, z. B. GlassFish.

Der Broker wird als Java EE Applikation realisiert.

Java EE stellt dabei einen Container zur Verfügung, in welchem die Komponente auf dem Application-Server betrieben wird. Die Architektur ist daher sehr einfach gehalten, da die Funktionalität in einem Java-Bean vorliegt.

5.3.2 Design-Pakete

Da die Broker-Komponente nicht realisiert werden konnte, liegen darum keine Design-Pakete vor.

6 Kommunikation

Die Softwarelösung ri4idd2dicom kommuniziert von einem Client (ri4idd2dicom-Client) über einen Broker (ri4idd2dicom-Broker), bzw. vom Broker mit dem DICOM-Server. Nachfolgender Abschnitt beschreibt die Kommunikation komponentenweise:

6.1 Client/Broker

Für die Kommunikation des Clients mit dem Broker wird eine JMS Queue verwendet. Diese JMS Queue wird durch einen eindeutigen Name (z. B. „ri4idd2dicom-queue“) gekennzeichnet, welche auf dem Application-Server bereitgestellt werden muss.

Die Kommunikation über die JMS-Queue ist nachrichtenbasiert und asynchron. Der Client sendet dabei seine Message an den Broker mit Inhalt (Absender, DICOM-Objekt) und dann ist für ihn das erledigt. Die Queue kann also als eine Art Sammelbehälter verstehen werden, welche von mehreren Produzenten (Clients) genutzt wird, aber nur einen Konsumenten (Broker) besitzen.

Ist der Zugriff auf die JMS-Queue durch den Client nicht möglich (arbeitet offline), dann werden die Daten lokal in der Applikation zwischengespeichert.

Der Broker liest periodisch aus der Queue und verarbeitet die Daten bei sich, bzw. wird informiert, wenn neue Nachrichten eingetroffen sind.

6.2 Broker/DICOM-Server

Für die Kommunikation des Brokers mit dem DICOM-Server wird das DICOM Applikationsprotokoll eingesetzt. Das DICOM Applikationsprotokoll setzt auf TCP/IP auf, verwendet aber zusätzliche protokollspezifische Angaben.

Hostname / IP-Adresse	**Port**	**AE Title**
localhost	11112	DCM4CHEE

Table 7: Verbindungsparameter für Kommunikation mit DICOM-Server

6.2.1 Verbindungsüberprüfung

Damit der Broker die Daten weiterleiten kann, muss dieser die Service Verfügbarkeit des DICOM-Servers überprüfen.

Dies geschieht über einen C-Echo Request, welche der Service Class User (SCU) an den Service Class Provider (SCP) sendet (zyklisches Prüfen der Verbindung).

SCU ---- testen der Verbindung mit: C-Echo-Rq -----> SCP

SCU <--- C-Echo-Rsp ---------- SCP

Ist eine Response des DICOM-Servers auf den C-Echo-Rq angekommen, dann weiss der Broker, dass die Verbindung richtig initialisiert wurde und eine Datenkommunikation auf DICOM-Ebene zu Stande kommt.

7 Prozesse und Threads

Nachfolgendes Kapitel gibt eine Übersicht der vorhandenen Prozesse/Threads:

7.1 Client

Durch den Einsatz des Swing Application Frameworks auf der Client Seite wird eine Umgebung zum Ausführen Threads innerhalb des Client-Prozesses durch dieses Framework vorgegeben.

Die Applikation wird in seinem eigenen Client-Prozess gestartet.

Für den Betrieb des GUIs ist ein SwingWorker Thread im Einsatz.

Für das Senden der DICOM-Daten wird ein entsprechender Task eingesetzt, welcher als Background-Thread realisiert wird, der auch blockieren kann (wie Netzwerk oder Datei IO).

7.2 Broker

Da die Broker-Komponente innerhalb eines Application-Servers betrieben wird ist der EJB-Container dafür verantwortlich, dass für die Lebensdauer des Broker seine Ressourcen zu Verfügung gestellt werden.

8 Systemeigenschaften

Dieser Abschnitt beschreibt die Systemeigenschaften dieser Softwarelösung, welcher für einen reibungslosen Betrieb vorausgesetzt wird:

8.1 Anforderungen Client

8.1.1 Hardware Voraussetzungen

Die Hardware Voraussetzungen für den Client entsprechen einer durchschnittlichen Computerhardware:

- 2 GHz CPU
- 512MB RAM
- 40 GB Festplatte
- Netzwerkkarte (Ethernet oder WLAN)
- Standard Desktop-Grafikkarte und Monitor

8.1.2 Software Voraussetzungen

- Microsoft Windows XP mit aktuellstem SP
- Java SE (Standard Edition) in der Version 6
- mindestens 100MB freier Festplattenspeicher

8.2 Anforderungen Broker

8.2.1 Hardware Voraussetzungen

Mide-Range Serverhardware, auch als virtuelle Maschine einsetzbar

- 2 GHz CPU
- 2GB RAM
- Festplatte (Raid) oder auch SAN Speicher
- Netzwerkkarte (Ethernet)

8.2.2 Softwarevoraussetzungen Broker

- Java EE 5 kompatibler Application-Server
  (z.B. den Open Source Application-Server Glassfish von Sun)
- JMS Queue

8.3 Grösse und Leistung

Einschränkungen der Applikation bezüglich Speicher, Leistung sind in dieser Version nicht festgelegt.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Betreiberdokumentation

21. Mai 2009

Abstract

Dieses Dokument richtet sich an den Betreiber und Entwickler dieser Softwarelösung. Inhaltliche Schwerpunkte dabei sind Installation und Konfiguration der Arbeit- und Entwicklungsumgebung.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Betreiberdokumentation_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
05.03.2009	Dokument erstellt
21.05.2009	Dokument in Version 2 überarbeitet
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument richtet sich an den Betreiber und Entwickler für diese Softwarelösung. Es beinhaltet alle notwendigen Informationen zum Betrieb (Installation und Konfiguration) der Arbeit-, bzw. Entwicklungsumgebung.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Dieses Dokument gliedert in die Bereiche Entwicklungsumgebung und Arbeitsumgebung.

2 Entwicklungsumgebung

Dieser Abschnitt ist für den Entwickler der Softwarelösung gedacht:

2.1 Arbeitsstation

Meine Arbeitsstation ist ein Lenovo W500 Notebook mit Ubuntu Linux 8.10 als Betriebssystem. Mit 4GB RAM Arbeitsspeicher, einer 320GB Festplatte und einer multi core CPU (Intel Core 2 Duo T9550, 2.66 GHz) stellt dieser die benötigte Leistung zur Verfügung und ermöglicht zudem ein ortsunabhängiges Arbeiten.

2.1.1 Betriebssystem

Mit der bekannten Linux Distribution Ubuntu[1] wird ein aktuelles und freies Betriebssystem angeboten. Ich verwende den Ubuntu Release 8.10 ("Intrepid Ibex") mit den aktuellen Kernel 2.6.27-x (Kernel Linux 2.6.27-11-generic). Als Arbeitsoberfläche dient der GNOME Desktop in der Version 2.24.x (GNOME 2.24.1).

Das Betriebssystem wird täglich auf Security-Updates, bzw. Anwendungsaktualisierungen geprüft. Mittels folgendem Konsolenbefehl kann dieser Vorgang jederzeit auch manuell ausgeführt werden:

```
$ sudo apt-get update && sudo apt-get dist-upgrade
```

2.1.2 Hardware

Folgende Hardware besitzt mein Notebook

Arbeitsspeicher:	4.0GB
Prozessor 0:	Intel(R) Core(TM)2 Duo CPU T9550 @ 2.66GHz
Prozessor 1:	Intel(R) Core(TM)2 Duo CPU T9550 @ 2.66GHz
Festplatte:	/dev/sda 320GB
Netzwerkadapter:	eth0, wlan0

2.1.3 Partitionierung der Festplatte

Device	Directory	Type	Total
/dev/sda1		swap	8GB
/dev/sda2	/	ext3	30GB
/dev/sda3	/opt	ext3	100GB
/dev/sda3	/home	ext3	182GB

2.2 Software

Folgende Software wurde zusätzlich auf dem Notebook installiert, welche für diese Studienarbeit relevant sind:

1 http://www.ubuntu.com

- Sun xVM VirtualBox 2.1.4
- Mozilla Firefox 3.0.10
- OpenOffice.org 3.1.0
- Wireshark 1.0.3
- Adobe Reader 8
- Tortoise SVNClient

2.3 Entwicklungstools

Als Entwicklungstools werden die im Studium eingesetzten und meist verbreiteten Softwaretools verwendet. Gegeben durch die modulare Bauweise, lassen sich spezielle Plugins für die Entwicklungsumgebung bei Bedarf nachinstallieren.

2.3.1 IDE

Als integrierte Entwicklungsumgebung wurden die zwei unten aufgeführten IDE's verwendet, welche beide auf der jeweilige Seite des Herstellers kostenlos heruntergeladen werden kann.

IDE Name	**Einsatzgebiet**
Eclipse 3.4 [2]	Programmierung, Refactoring, Maven-Integration
NetBeans 6.5.1 [3]	Programmierung, besitzt sehr guten GUI-Editor (Matisse)

Table 1: Entwicklungsumgebung (IDE)

Gerade beim erstellen des GUI's ist der integrierte GUI-Editor von NetBeans sehr hilfreich. Leider ist das Zusammenspiel mit Eclipse nicht gut gelöst, welches sich beim Ändern der GUI-Klassen bemerkbar macht. Dies macht sich bemerkbar durch die fehlende Synchronisation der GUI-Komponenten Metadaten, welche vom GUI-Editor erstellt werden.

Wird beispielsweise in Eclipse ein Refactoring am Client durchgeführt, welche GUI-Komponenten beinhalten, dann lassen sich dir GUI-Komponenten im GUI-Editor nicht mehr richtig öffnen.

Folgende Empfehlung kann daher gegeben werden:

Werden Änderungen am GUI vorgenommen, dann soll mit NetBeans gearbeitet werden. Für Änderungen ausserhalb des GUI's kann mit Eclipse eingesetzt werden.

2.4 Datenbank

Für die Datenspeicherung seitens des Clients wird lokale eine HSQLDB[4] (Hyper Structured Query Language Database) eingesetzt. Damit diese im Client-Projekt verwendet werden kann sind folgende Einstellungen vorzunehmen:

2 http://www.eclipse.org/
3 http://www.netbeans.org/
4 http://hsqldb.org/

Name:	HSQLDB (Embedded)
User Name:	sa
Password:	[leer]
JDBC URL:	jdbc:hsqldb:file:db/ri4idd2dicomdb
Selected Schema:	PUBLIC

Table 2: HSQLDB Datenbankeigenschaften Client

Die eingesetzte HSQLDB Datenbank wird beim Starten des Clients geladen und beim Schliessen des Clients wieder beendet, welche gemäss der HSQLDB-Dokumentation mit dem Begriff In-Prozess-Modus bezeichnet wird.

Die Datenbank wird im File-Modus betrieben, d.h. innerhalb des Datenbankspeicherortes werden die Daten zum Betrieb der DB in unterschiedliche Daten abgelegt (z. B. ri4idd2dicomdb.properties, ri4idd2dicomdb.script, ri4idd2dicomdb.log)

Wenn der Client (ri4idd2dicom-client) auf der Arbeitsstation des Endanwenders betrieben wird, dann muss die Datenbank im Dateisystem im Unterordner „db“ vorliegen, damit die Datenbank überhaupt gefunden, bzw. gelesen werden kann.

2.4.1 Verwalten der Datenbankzustände

Für das Arbeiten mit der Datenbank müssen diverse Datenbankzustände gegeben werden. Die hier aufgeführten Zustände beschreiben den Inhalt der Daten, welche auf der Datenbank vorliegen.

Zustand	**Beschreibung**
Entwicklung	Enthält Testdaten
Deployment	Enthält vorgegebene Daten, welche für den Betrieb der Applikation vorausgesetzt wird.
Produktiv	Enthält live Daten

Table 3: Datenbankzustand

Arbeitet der Entwickler an dem Client, verwendet er die Datenbank im Entwicklungs-Zustand. Wir die Datenbank auf die Arbeitsstation des Benutzers verteilt, dann muss die Datenbank mit dem Deployment-Zustand verwendet werden. In dem Moment, in welchem der Benutzer mit der Applikation zum ersten mal arbeitet, wechselt der Zustand der Datenbank in den Produktiv-Zustand.

Für das Testen der Datenbank kann die Datenbank im Entwicklungszustand verwendet, bzw. innerhalb der Testmethoden mit den nötigen Daten aufgefüllt und gelöscht werden.

3 Arbeitsumgebung

Dieser Abschnitt richtet sich an den Betreiber der Infrastruktur dieser Softwarelösung

3.1 Virtuelle Arbeitsumgebung

Mit der freien Desktop Virtualisierungslösung Sun xVM VirtualBox[5] steht dem Betreiber eine umfangreiche Softwarelösung zum Erstellen virtueller Maschinen zur Verfügung.

Ich habe mich für den Einsatz einer virtuellen Infrastruktur entschieden, da die virtuellen Gäste in einer getrennten Arbeitsumgebung von meinem System arbeiten sollen. Da die virtuellen Gäste lokal jeweils in einer Imagedatei gespeichert werden, ermöglicht es diese einfach zu sichern und bei Bedarf auf eine andere Maschine zu kopieren. Daneben lassen sich mit VirtualBox Snapshots anfertigen, um den aktuellen Zustand einer virtuellen Maschine festzuhalten.

3.2 Sun xVM VirtualBox

Am 16.02.2009 wurde die Version 2.1.4 von VirtualBox veröffentlicht. Diese Version kann auf der Produktwebseite unter dem Bereich Download[6] für das jeweilig eingesetzte Betriebssystem heruntergeladen werden.

Für mein Betriebssystem Ubuntu 8.10 ("Intrepid Ibex") wurde die passende Linux Version von VirtualBox 2.1.4 heruntergeladen.

3.2.1 Installation

Mittels folgendem Befehl wird VirtualBox per Konsole installiert:

```
$ sudo dpkg -i /home/myusername/virtualbox-2.1_2.1.4-42893_Ubuntu_intrepid_i386.deb
```

3.3 Konfiguration

3.3.1 Allgemeine Informationen

Es werden drei virtuelle Maschinen erstellt, welche je nach Gebrauch alle gleichzeitig laufen können:

5 http://www.virtualbox.org/
6 http://www.virtualbox.org/wiki/Downloads

Name	Testserver1	Testserver2	Windows XP Dev
Aufgaben	DICOM-Server	Build-Server Glassfish	Entwicklungsumgebung für Windows Treiberentwicklung
Betriebssystem	Ubuntu Server 8.04.2	Ubuntu Server 8.04.2	Windows XP SP3
Festplatte	10GB /dev/sda1 swap (1GB) /dev/sda2 ext3 (9GB)	10GB /dev/sda1 swap (1GB) /dev/sda2 ext3 (9GB)	10GB c:\ (10GB)
Netzwerkadapter	1	1	1
IP Adresse	192.168.168.11	192.168.168.12	192.168.168.21

Table 4: Übersicht der eingerichteten virtuellen Maschinen

3.3.2 Schritt für Schritt-Anleitung zum Erstellen einer virtuellen Maschine

- Sun xVM VirtualBox starten
- Im Anwendungsfenster mittels Button „New“ den Wizard starten -> Next
- Name für VM vergeben, z.B. „Testserver1“ und OS Type entsprechend auswählen: Operating System: „Linux“ und Version: „Ubuntu“ -> Next
- Selektieren des zu verwendenden Arbeitsspeicher: Base Memory Size: „512“ MB -> Next
- Virtuelle Festplatte erstellen -> Button „New...“ -> Next -> Storage Type: „Dynamically expanding storage“ -> Next -> Location: „Testserver1“, Size: „10.00 GB“ -> Finish -> Next
- Zusammenfassung (Summary) anschauen -> Finish

Die virtuelle Maschine wurde soeben in VirtualBox erstellt und findet sich links in der Inventarübersicht. Als nächstes wir die virtuelle Maschine noch weiter konfiguriert -> VM selektieren und mittels Button „Settings“ die Feineinstellungen an der VM vornehmen:

- General
  - Tab: General
    - Video Memory Size: 8MB
  - Tab: Advanced
    - Boot Order: Floppy deselektieren und nach unten verschieben
    - Extended Features: alles aktivieren
  - Tab: Description
    - [-> hier die spezifische Beschreibung zum Server reinschreiben]
- Hard Disks
  - Enable SATA Controller: selektieren
- Network

- Tab: Adapter 1
  - Enable Network Adapter: selektieren
  - Adapter Type: Intel PRO/1000 T-Server (82543GC)
  - Attached to: Host Interface
  - Host Interfaces: tap0 selektieren
- USB
  - Enable USB Controller: selektieren
  - Enable USB 2.0 (EHCI) Controller: selektieren

3.3.2.1 Netzwerkkonfiguration

Damit die virtuellen Maschinen untereinander kommunizieren können, werden diese in ein Netzwerk integriert. Folgende Anforderungen werden dabei gestellt:

- VMs sollen Zugriff auf das Internet haben
- VMs sind in einem separaten Subnetz und können untereinander kommunizieren
- VMs können mit Host kommunizieren, bzw. Host hat Netzwerkverbindung zu den VMs

Damit die VMs untereinander Daten austauschen können, muss als erstes ein virtuelles Netzwerk Gerät (TAP)[7] eingerichtet werden,. Auf diesem Interface werden für jeder VM die entsprechende Netzwerkkonfiguration vorgenommen. Diese Lösung erlaubt es den VMs über das gleiche Netzwerk zu kommunizieren, während dem die Wireless-Netzwerkkarte auf dem Host als Gateway zum Internet hin dient.

Ab der Version 2.1.x von VirtualBox wurde neu die Option „Host Interface networking“ hinzugefügt. Dies erlaubt einem die Host Maschine als Netzwerk Gateway für die Gast Maschine auszuwählen.

Um den VMs Internetzugriff zu ermöglichen, muss in den Host Iptable Regeln das IP Masquerading eingestellt werden.

Die IP Adressen auf den VMs sind so zu konfigurieren, dass diese aus dem privaten IP Adressbereich kommen und im gleichen Subnetz sich befinden.

3.3.2.2 Schritt für Schritt Anleitung für die Konfiguration des Linux Hosts

- Sicherstellen ob die uml-utilities installiert sind

```
$ sudo apt-get install uml-utilities
```

- Erstellen einer virtuellen Netzwerkschnittstelle

```
$ sudo tunctl -t tap0 -u $USER
```

(wobei $USER der Benutzer ist, welche die VirtualBox Sitzung startet)

- Sicherstellen, ob Benutzer in der vboxusers Gruppe ist:

7 http://vtun.sourceforge.net/tun/

```
$ cat /etc/group
```

(für Gruppeneintrag „vboxusers“ nach schauen und überprüfen, dass $USER Mitglied dieser Gruppe ist)

- Sicherstellen ob die vboxusers Gruppe Zugriffsrechte auf die tun devices hat:

```
$ sudo chgrp vboxusers /dev/net/tun
$ sudo chmod 660 /dev/net/tun
```

- Netzwerkschnittstelle aktivieren und IP Adresse zuweisen
  - Sicherstellen, dass die IP-Adresse und Netzwerkmaske nicht anderswo in Gebrauch ist; ich verwendete 192.168.168.1, welche sich nicht störte mit anderen.

```
$ sudo ifconfig tap0 192.168.168.1
```

- NAT forwarding einrichten:

```
$ sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

(Austauschen der passenden Netzwerkschnittstelle - hier wlan0 - gemäss denjenigen auf der lokalen Maschine.)

```
$ sudo sysctl -w net.ipv4.ip_forward=1
```

- Auf dem Guest Netzwerkkonfiguration vornehmen -> statische IP Netzwerkadresse verwenden!

Einstellungen \ VM	**Testserver1**	**Testserver2**	**Windows XP Dev**
IP Adresse:	192.168.168.11	192.168.168.12	192.168.168.21
Subnetzmaske:	255.255.255.0	255.255.255.0	255.255.255.0
Default Gateway	192.168.168.1	192.168.168.1	192.168.168.1

Table 5: IP-Konfiguration der VMs

- DNS Server zuweisen basierend auf den Einstellungen, welche auf dem Host verwendet werden.

```
$ sudo nano /etc/resolv.conf
```

Einen einzigen Nachteil hat das Ganze: das soeben erstellte TAP Device wird verschwinden, wenn die Hostmaschine neu gestartet wird. Um diese Problematik zu lösen, wird folgende Zeilen in der Datei /etc/rc.local hinzugefügt.

```
$ sudo nano /etc/rc.local

echo -n "Setting up tap0 interface..."
tunctl -t tap0 -u myusername
ifconfig tap0 192.168.168.1
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
sysctl -w net.ipv4.ip_forward=1
echo "DONE!"
```

Diese Konfiguration stellt sicher, dass das TAP Device eingerichtet ist und dass auch das IP Masquerading beim Systemstart aktiviert ist.

Zum Schluss muss noch die Hostdatei angepasst werden um den Servername hinzuzufügen:

```
$ sudo nano /etc/hosts

127.0.0.1     localhost
127.0.0.1     Testserver2
```

Die Idee zu dieser Konfiguration wurde aus dem Blogeintrag „VirtualBox Networking“ entnommen:

http://weierophinney.net/matthew/archives/205-VirtualBox-Networking.html [Abruf: 2009-05-25]

3.3.3 Installation Ubuntu Server

- Iso Image-Datei von Ubuntu-Server 8.04.2 auf Webseite herunterladen.
- Virtuelle Maschine erstellen
- VM starten und iso-Datei im CD-ROM mounten.
- F2 drücken um Sprache umzustellen -> English -> ENTER
- F3 drücken um Tastaturlayout umzustellen -> Swiss German -> ENTER
- Install Ubuntu Server auswählen -> ENTER
- English-Englisch als Sprache auswählen -> ENTER
- Country auswählen -> other -> ENTER -> -- Europe – Switzerland auswählen -> ENTER
- Hostname: trac -> ENTER
- Partition disks -> Manual -> ENTER
- Disk selektieren [pri/log 10.7 GB FREE SPACE] -> ENTER -> Create a new partition -> ENTER -> New partition size: 1GB -> ENTER -> Primary -> ENTER -> Beginning -> ENTER -> Use as: swap area -> ENTER -> Done setting up the partition -> ENTER
- Disk selektieren [pri/log 9.7 GB FREE SPACE] -> ENTER -> Create a new partition -> New partition size: 9.7GB -> ENTER -> Primary -> ENTER -> Bootable flag: ENTER -> Done setting up the partition -> ENTER
- Finish partitioning and write changes to disk -> ENTER -> Write the changes to disks -> Yes -> ENTER
- Full name for the new User: Peter Muster -> ENTER
- Username for your account: pmuster -> ENTER
- Choose a password for the new user: ********** -> ENTER (und nochmals bestätigen) -> ENTER
- HTTP proxy information (blank for none): -> ENTER
- Choose software to install: [x] OpenSSH server -> [x] PostgreSQL database -> Continue
- Finish the installation -> Continue
- CD-ROM wieder entfernen -> System wird neu gestartet
- Anmelden mit Benutername und Passwort
- IP-Adresse ändern:

```
$ sudo nano /etc/network/interfaces
```

- eth0 interface statische IP-Adresse konfigurieren:

```
# The primary network interface
auto eth0
iface eth0 inet static
```

```
        address 192.168.168.11
        netmask 255.255.255.0
        network 192.168.168.0
        broadcast 192.168.168.255
        gateway 192.168.168.1

$ sudo /etc/init.d/networking restart
```

- Ubuntu aktualisieren:

```
$ sudo apt-get update && sudo apt-get dist-upgrade

$ sudo apt-get autoclean

$ sudo apt-get autoremove
```

- Server neu starten:

```
$ sudo reboot now
```

3.3.4 Installation Guest Additions unter Linux

Es empfiehlt sich nach dem Installieren der virtuellen Maschine die entsprechenden Guest Additions (sind spezifische Treiber für das Betriebssystem) zu installieren.

Als erstes müssen die Build-Komponenten und Linux Header-Dateien heruntergeladen werden:

```
$ sudo apt-get install build-essential linux-headers-$(uname -r)
```

In der Menüliste des Guest-Fensters die Treiber-CD laden:

Devices -> Install Guest Additions...

```
$ sudo mount /cdrom

$ cd /media/cdrom

$ sudo ./VboxLinuxAdditions-x86.run
```

Ist die Installation erfolgreich abgeschlossen, muss die virtuelle Maschine neu gestartet werden:

```
$ sudo reboot now
```

Installation ist abgeschlossen

3.3.5 Entwicklungsumgebung für Treiber

Für die Treiberentwicklung wird wiederum eine virtuelle Maschine erstellt. Diese VM hat den Namen „Windows XP Dev“ und als Betriebssystem wird ein Microsoft Windows XP Professional mit SP3 inkl. Windows Update, verwendet.

Partitionierung: 10GB Harddisk NTFS formatiert

IP: 192.168.168.21, 255.255.255.0, Default gateway: 192.168.168.1

Preferred DNS server: 192.168.1.1, Alternate DNS server: 192.168.10.1

Software installiert:

- Sun xVM VirtualBox Guest Additions 2.1.4

- Avira AntiVir Personal – Free Antivirus[8]
- Microsoft Visual Studio 2008
- Windows Driver Kit (WDK)[9] für Treiberentwicklung

3.3.5.1 Installation WDK

Das WDKs muss über die MSDN subscriber downloads heruntergeladen werden.

Dateiname: en_windows_driver_kit_6001.18002_sp1_x86_x64_ia64_284919.iso

- ISO-Datei mit Laufwerk in VM mappen
- Autostart der CD
- Microsoft Windows Driver Kit Release 6.1.6001.18002 Configuration Manager – Not Configured
    - Microsoft WDK Component Configuration:
        - Full Development Environment: selektieren
        - Microsoft Windows Debuggers: selektieren
        - Device Simulation Framework: selektieren
    - Button OK
- Installation Location Dialog
    - Install Path: C:\WinDDK\6001.18002\
    - Button OK klicken
- End-User License Agreement: I Agree selektieren, Button OK klicken

Installationsdauer beträgt ca. 30Min und verbraucht dabei ca. 1.5GB Festplattenspeicher

8 http://www.free-av.de/en/download/index.html
9 http://en.wikipedia.org/wiki/Windows_Driver_Kit

3.4 Glassfish Application-Server

Für das Betreiben der Broker-Komponente wurde ein Glassfish Application-Server von Sun installiert.

Quelle Installationsanleitung:

http://weblogs.java.net/blog/cayhorstmann/archive/2006/07/installing_glas.html [Abruf: 2009-05-21]

http://weblogs.java.net/blog/cayhorstmann/ [Abruf: 2009-05-21]

JMS:

http://today.java.net/pub/a/today/2008/01/22/jms-messaging-using-glassfish.html [Abruf: 2009-05-21]

3.4.1 Installation

- Verbindung zu Server herstellen:

```
$ ssh 192.168.10.12
```

- Server aktualisieren:

```
$ sudo apt-get update && sudo apt-get dist-upgrade
```

- Java6 JDK installieren:

```
$ sudo apt-get install sun-java6-jdk
```

- Variable JAVA_HOME setzen und folgende Zeile hinzufügen:

```
$ sudo nano /etc/environment

JAVA_HOME="/usr/lib/jvm/java-6-sun/jre"
```

- Download Glassfish:
  Webseite für passende Version:
  https://glassfish.dev.java.net/downloads/v2.1-b60e.html [Abruf: 2009-05-21]
- Datei ins home-Verzeichnis downloaden:

```
$ wget http://java.net/download/javaee5/v2.1_branch/promoted/Linux/glassfish-
installer-v2.1-b60e-linux.jar
```

- Datei entpacken:

```
$ java -Xmx256m -jar glassfish-installer-v2.1-b60e-linux.jar
```

- Heruntergeladene Datei löschen:

```
$ rm glassfish-installer-v2.1-b60e-linux.jar
```

- Verzeichnis nach /opt verschieben:

```
$ sudo mv glassfish /opt/glassfish-v2.1-b60e
```

- Symlink erstellen:

```
$ sudo ln -s /opt/glassfish-v2.1-b60e /opt/glassfish
```

- Berechtigungen anpassen:

```
$ sudo chown -R root:root glassfish-v2.1-b60e/
```

- ant ausführbar machen:

```
$ sudo chmod +x /opt/glassfish/lib/ant/bin/ant
```

- Setup starten:

```
$ sudo /opt/glassfish/lib/ant/bin/ant -f /opt/glassfish/setup.xml
```

- Startupskript erstellen:

```
$ sudo nano /etc/init.d/glassfish
```

- Unten stehender Inhalt (wurde aus Vorlage genommen und angepasst) in Datei /etc/init.d/glassfish einfügen:

```
#! /bin/sh
### BEGIN INIT INFO
# Provides:          glassfish
# Required-Start:    $local_fs $remote_fs
# Required-Stop:     $local_fs $remote_fs
# Default-Start:     2 3 4 5
# Default-Stop:      S 0 1 6
# Short-Description: glassfish initscript
# Description:       A simple initscript for the glassfish app server
### END INIT INFO
#
# Author:            Cay S. Horstmann (http://horstmann.com)
#

set -e

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/glassfish/bin
DESC="Sun GlassFish Enterprise Server v2.1"
NAME=glassfish
ASADMIN=asadmin
PIDFILE=/var/run/$NAME.pid
SCRIPTNAME=/etc/init.d/$NAME

# Gracefully exit if the package has been removed.
test -x $DAEMON || exit 0

# Read config file if it is present.
#if [ -r /etc/default/$NAME ]
#then
#       . /etc/default/$NAME
#fi

#
#       Function that starts the daemon/service.
#
d_start() {
        $ASADMIN start-domain \
                || echo -n " already running"
}

#
#       Function that stops the daemon/service.
#
d_stop() {
        $ASADMIN stop-domain \
                || echo -n " not running"
}

case "$1" in
  start)
```

```
        echo -n "Starting $DESC: $NAME"
        d_start
        echo "."
        ;;
  stop)
        echo -n "Stopping $DESC: $NAME"
        d_stop
        echo "."
        ;;
  reload|restart|force-reload)
        echo -n "Restarting $DESC: $NAME"
        d_stop
        sleep 10
        d_start
        echo "."
        ;;
  *)
        echo "Usage: $SCRIPTNAME {start|stop|restart|force-reload}" >&2
        exit 3
        ;;
esac

exit 0


Skript ausführbar machen:
$ sudo chmod +x /etc/init.d/glassfish
Skript für automatischen Systemstart vorsehen:
$ sudo update-rc.d glassfish defaults
```

3.4.2 Administrative Arbeiten

- Verbinden auf Application-Server Admin Konsole per Webbrowser: http://192.168.168.12:4848
- Admin Passwort ändern: (funktioniert nur, wenn Server gestartet ist)

```
$ /opt/glassfish/bin/asadmin change-admin-password --user admin
Please enter the old admin password>adminadmin
Please enter the new admin password>mysecret
Please enter the new admin password again>mysecret
```

- Server starten (manuell):

```
$ sudo /opt/glassfish/bin/asadmin start-domain domain1
```

- Server stoppen (manuell):

```
$ sudo /opt/glassfish/bin/asadmin stop-domain domain1
```

3.4.2.1 PostgreSQL

Damit der Application-Server mit einer Relationalen Datenbank kommunizieren kann, braucht dieser einen passenden JDBC-Treiber.

- Aktuellster JDBC-Treiber[10] von PostgreSQL für den Glassfish herunterladen:

```
$ wget http://jdbc.postgresql.org/download/postgresql-8.3-604.jdbc4.jar
```

- Datenbanktreiber nach /opt/glassfish/domains/domain1/lib verschieben:

```
$ sudo mv postgresql-8.3-604.jdbc4.jar /opt/glassfish/domains/domain1/lib
```

10 http://jdbc.postgresql.org/download.html

- Rechte ändern:

```
$ sudo chown root:root /opt/glassfish/domains/domain1/lib/postgresql-8.3-
604.jdbc4.jar
```

3.5 DICOM Server

dcm4che[11] ist eine Sammlung von Open Source Anwendungen und Utilities für Unternehmen in der Medizinbranche. Die Anwendungen wurden in Java entwickelt und basieren auf dem JDK 1.4 oder höher.

Für eine umfassende Übersicht verweise ich auf die Startseite[12] des dcm4chee-2.x.

3.5.1 Systemübersicht

Siehe Webseite von dcm4che.org:
http://www.dcm4che.org/confluence/display/ee2/System+Overview [Abruf: 2009-05-21]

3.5.2 Installation

Die Installation wurde gemäss der offiziellen Installationsanleitung[13] durchgeführt.

Nachfolgend werden die wichtigsten Schritte erklärt, welche speziell für diese Umgebung relevant sind:

- Java6 JDK installieren - beim Setup muss man den Lizenzbedingungen zustimmen:

```
$ sudo apt-get install sun-java6-jdk
```

- Download des binary distribution package von dcm4chee auf sourceforge.net[14]:
  Aktuellste Version: 2.14.2 → dcm4chee-psql-2.14.2.zip

```
$ wget http://downloads.sourceforge.net/dcm4che/dcm4chee-psql-2.14.2.zip?
use_mirror=switch
```

- Programm unzip installieren um Dateien zu entpacken:

```
$ sudo apt-get install unzip
```

Zip-Datei extrahieren und anschliessend ins /opt Verzeichnis verschieben:

```
$ unzip dcm4chee-psql-2.14.2.zip

$ sudo mv dcm4chee-psql-2.14.2 /opt
```

3.5.2.1 Installation von JBoss-4.2.3.GA

Server seitig wird ein JBoss Application-Server vorausgesetzt, welcher auf der Webseite[15] des Herstellers heruntergeladen werden kann.

Eingesetzte Version: 4.2.3.GA → jboss-4.2.3.GA-jdk6.zip

- JBoss Binary Datei herunterladen:

```
$ wget http://switch.dl.sourceforge.net/sourceforge/jboss/jboss-4.2.3.GA-jdk6.zip
```

- Tipp: Die Zip-Datei kann auch mittels dem JDK jar Tool extrahiert werden:

11 http://www.dcm4che.org/
12 http://www.dcm4che.org/confluence/display/ee2/Home
13 http://www.dcm4che.org/confluence/display/ee2/Installation
14 http://sourceforge.net/project/showfiles.php?group_id=37982&package_id=195374
15 http://www.jboss.org/jbossas/downloads/

```
$ jar -xvf jboss-4.2.3.GA-jdk6.zip
```

- Eine Verzeichnis mit Name jboss-4.2.3.GA ist nun vorhanden, welches in das /opt Verzeichnis verschoben werden muss:

```
sudo mv jboss-4.2.3.GA /opt
```

- Als nächstes muss die JBOSS_HOME variable (für Linux) angepasst werden: siehe http://www.jboss.org/file-access/default/members/jbossas/freezone/docs/Installation_Guide/4/html/setting_JBOSS_HOME_linux.html [Abruf: 2009-05-21]
- weiter nach Punkt 3. aus der offiziellen Installationsanleitung
- Ins folgendes Verzeichnis wechseln

```
$ cd /opt/dcm4chee-psql-2.14.2/bin
```

- Mit nachfolgendem Befehl Skriptdatei ausführen um Installation zu starten:

```
$ sudo ./install_jboss.sh /opt/jboss-4.2.3.GA/
```

3.5.2.2 PostgreSQL Datenbank installieren

Ist Punkt 4. aus der offiziellen Installationsanleitung.

Für weiterführende Informationen kann auf folgenden Link verwiesen werden: http://www.dcm4che.org/confluence/display/ee2/PostgreSQL [Abruf: 2009-05-21]

Da PostgreSQL bei der Installation des Servers defaultmässig installiert wurde, muss diese Datenbank nur noch konfiguriert werden:

- In PostgreSQL Verzeichnis wechseln:

```
cd /etc/postgresql/8.3/main
```

- PostgreSQL Konfigurationsdatei editieren und Eintrag Connection Settings anpassen:

```
sudo nano postgresql.conf

listen _address = '*'
```

- Einstellen der Zugriffsbeschränkung auf IP-Basis[16]:

```
$ sudo nano pg_hba.conf

# IPv4 local connections:
host    all         all         192.168.10.0/24       md5
```

- Ändern des Passwortes des Postgres-Benutzers auf Seiten der Datenbank

```
$ sudo -u postgres psql postgres

postgres=# ALTER USER postgres WITH ENCRYPTED PASSWORD 'postgres';
postgres=# \q
```

- Passwort von Benutzer ändern (Passwort: „postgres“)

```
$ sudo passwd postgres
```

- Erstellen der Datenbank pacsdb durch die mitgelieferten SQL-Dateien:

16 http://www.postgresql.org/docs/8.3/static/client-authentication.html

```
$ sudo -u postgres psql pacsdb -f /opt/dcm4chee-psql-2.14.2/sql/create.psql

$ sudo -u postgres psql pacsdb
```

- Anpassen der Konfiguration, d.h. Passwort des Benutzernamen “postgres” noch eingeben: <password>postgres<postgres>

```
$ sudo nano /opt/dcm4chee-psql-2.14.2/server/default/deploy/pacs-postgres-ds.xml
```

3.5.2.3 Abschlussarbeiten

- Punkt 5. aus der offiziellen Installationsanleitung kann ausgelassen werden.
- Variable JAVA_HOME setzen und folgende Zeile hinzufügen:

```
$ sudo nano /etc/environment

JAVA_HOME="/usr/lib/jvm/java-6-sun/jre"
```

- Punkt 7. aus der Installationsanleitung ist optional
- Ändern von -Xmx512m auf -Xmx256m

```
$ sudo nano /opt/dcm4chee-psql-2.14.2/bin/run.conf

#
# Specify options to pass to the Java VM.
#
if [ "x$JAVA_OPTS" = "x" ]; then
   JAVA_OPTS="-Xms128m -Xmx256m -Dsun.rmi.dgc.client.gcInterval=3600000
-Dsun.rmi.dgc.server.gcInterval=3600000"
fi
```

- Server starten um Installation zu testen (Punkt 9.)

```
$ cd /opt/dcm4chee-psql-2.14.2/bin
$ ./run.sh
```

Beim Starten des Servers tauchte ein Fehler auf. Durch entsprechende Recherche im Internet konnte in einem Forum[17] die passende Lösung gefunden werden. Lösung: Anpassen der Hostdatei (Eingabe des Hostnamen zur local host IP-Adresse):

```
$ sudo nano /etc/hosts

127.0.0.1 Testserver
```

- Anmelden an Webschnittstelle (Punkt 10. aus der offiziellen Installationsanleitung)
- Via Webbrowser auf Server verbinden: http://192.168.168.11:8080/dcm4chee-web/
  - Loginname: admin, Password: admin
- Als Service konfigurieren:

```
cd /opt
cp /opt/dcm4chee-psql-2.14.2/bin/dcm4chee_init_redhat.sh /etc/init.d
sudo nano /etc/init.d/dcm4chee_init_redhat.sh
```

- Server starten:

```
./dcm4chee_init_redhat.sh start
```

17 http://www.coderanch.com/t/91015/JBoss/Error-starting-Jboss-GA-ubuntu

- Server beenden:

```
./dcm4chee_init_redhat.sh stop
```

3.5.3 Administration des DICOM-Servers

Wurde der DICOM-Server entsprechend aufgesetzt, so muss dieser während seiner Betriebsdauer entsprechend administriert werden können. Die nachfolgenden Informationen geben einen kurzen Einstieg in diese Arbeiten.

3.5.3.1 Server starten

Wird als VM-Image betrieben (Name: Testserver1)

- Starten des Servers über die Sun xVM VirtualBox
- Konsolenfenster öffnen auf Client und mit Testserver1 verbinden:

```
ssh 192.168.168.11
```

- Anmelden mit Username und Passwort (fvetter, mysecretpw)
- DICOM-Server starten

```
$ cd/opt/dcm4chee-psql-2.14.2/bin

$ ./run.sh
```

3.5.3.2 Anmeldung an Server via Webschnittstelle

Der eingesetzte DICOM-Server bietet eine grafische Benutzerschnittstelle, welche über eine Web-Browser Verbindung eingesehen werden kann.

- Web-Browser starten
- URL eingeben: http://192.168.168.11:8080/dcm4chee-web/
- Loginname: admin
- Passwort: admin

3.5.3.3 Suchen von Daten

- Im Hauptmenü auf den Tab Folder wechseln
- Suchoptionen anpassen, z.B. aktivieren der Option w/o studies
- Klicken auf den Button mit dem Feldstecher „New Search“

System Zeigt eine Übersicht aller Patienten (Patient), sortiert nach Studie (Study) und Serien (Series) an.

3.5.3.4 Einstellungen AE Name

Damit später für die Service Class User (SCU), also all jene welche den DICOM-Server in Anspruch nehmen, die Anwendung spezifiziert werden kann, müssen auf dem DICOM-Server diese Einstellungen vorgenommen werden.

AE, bezeichnet die Applikationen und muss einen aussagekräftigen Namen besitzen.

Ein Beispiel eines AE Namens wäre „DCM4CHEE“, also der Name des DICOM-Servers.

- Web-Browser starten
- URL eingeben: http://192.168.168.11:8080/dcm4chee-web/ae.m
- Einstellungen zum AE Name serverseitig vornehmen

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Benutzerdokumentation

21. Mai 2009

Abstract

Dieses Dokument beschreibt die Benutzerdokumentation dieser Softwarelösung. Inhalt dabei ist die Installation sowie die Bedienungsanleitung.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Benutzerdokumentation_v2.0 .odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
21.05.2009	Dokument erstellt
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument beschreibt die Benutzeranleitung für diese Softwarelösung.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Diese Dokument gliedert sich in eine Installationsanleitung sowie eine Bedienungsanleitung.

2 Installationsanleitung

Dieses Kapitel beschreibt die Installation des Clients (ri4idd2dicom-Client), welcher auf der Arbeitsstation des Benutzers betrieben wird.

2.1 Anforderungen Client

2.1.1 Hardware Voraussetzungen

Die Hardware Voraussetzungen für den Betrieb des Clients (ri4idd2dicom-Client) setzen eine durchschnittlichen Computerhardware voraus:

- 2 GHz CPU
- 512MB RAM Arbeitsspeicher
- 40 GB Festplatte
- Netzwerkkarte (Ethernet oder WLAN)
- Standard Desktop-Grafikkarte und Monitor

2.1.2 Software Voraussetzungen

Die Softwarelösung ist eine Java Desktop-Applikation, welche für den Betrieb eine Java Laufzeitumgebung voraussetzt:

- Microsoft Windows XP mit aktuellstem SP
- Java SE (Standard Edition) in der Version 6
- mindestens 100MB freier Festplattenspeicher

2.2 Installation Client

Auf der beigelegten CD-ROM, welche das komplette SVN-Repository beinhaltet, ist ein Ordner *deploy* vorhanden. Dieser beinhaltet den aktuellsten Software-Build des Clients.

Damit der Client auf der Arbeitsstation des Benutzers betrieben werden kann, müssen folgende Schritte durchgeführt werden:

1. Kopieren des Unterverzeichnisses *ri4idd2dicom-client-1.0* aus dem *deploy* Ordners von der CD.ROM ins Programmverzeichnis des Clients.
   z.B. für Windows Betriebssystems nach C:\Programme\
2. Sicherstellen, dass Benutzer Schreibrechte in dieses Verzeichnis (z.B. C:\Programme\ ri4idd2dicom-client-1.0) besitzt.

2.2.1 Verknüpfung erstellen

Damit der Client (ri4idd2dicom-Client) durch den Benutzer über eine Verknüpfung auf dem Desktop gestartet werden kann, müssen folgende Einstellungen durchgeführt werden.

1. Wechseln Sie auf ihren Desktop
2. Mittels rechter Maustaste über das Kontextmenü einen Neuen Shortcut erstellen

(New -> Shortcut) Ein Wizard wird gestartet.

3. Gebe Sie „java.exe“ ein, wenn Sie nach dem Ziel gefragt werden
4. Benennen Sie den Shortcut mit „ri4idd2dicom-Client“ und beenden Sie anschliessend den Wizard.
5. Auf dem Desktop wurde ein entsprechendes Icon erstellt.
6. Mittels rechter Maustaste muss der Shortcut noch auf ihre Umgebung angepasst werden: (siehe unten stehende Abbildung: Illustration 1: Shortcut für Starten des Clients)

   Ergänzen Sie die Zeile Target (Ziel) mit folgendem Inhalt nach java.exe:
   -jar ri4idd2dicom-client-1.0-SNAPSHOT-jar-with-dependencies.jar

   Ergänzen Sie die Zeile Start in mit dem Pfad, in welchem Sie den Client lokal abgespeichert haben: „C:\Programme\ri4idd2dicom-client-1.0“

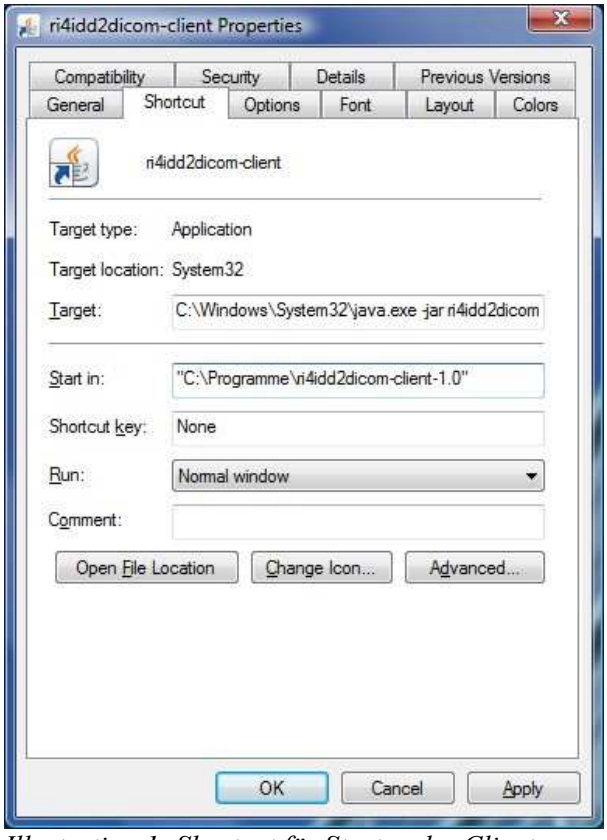


Illustration 1: Shortcut für Starten des Clients

7. Die Installation ist soweit abgeschlossen. Sie können den Client nun über den erstellten Shortcut starten.

3 Bedienungsanleitung

Dieses Kapitel beschreibt die Bedienung des Clients (ri4idd2dicom-Client). Es wird vorausgesetzt, dass der Client wie unter Kapitel 2 Installationsanleitung auf der Arbeitsstation des Benutzers entsprechend installiert ist.

3.1 Client starten

Der Client kann via Doppelklick auf das Desktop-Icon gestartet werden:

Illustration 2: Desktop-Icon Client

3.2 Grafische Benutzeroberfläche

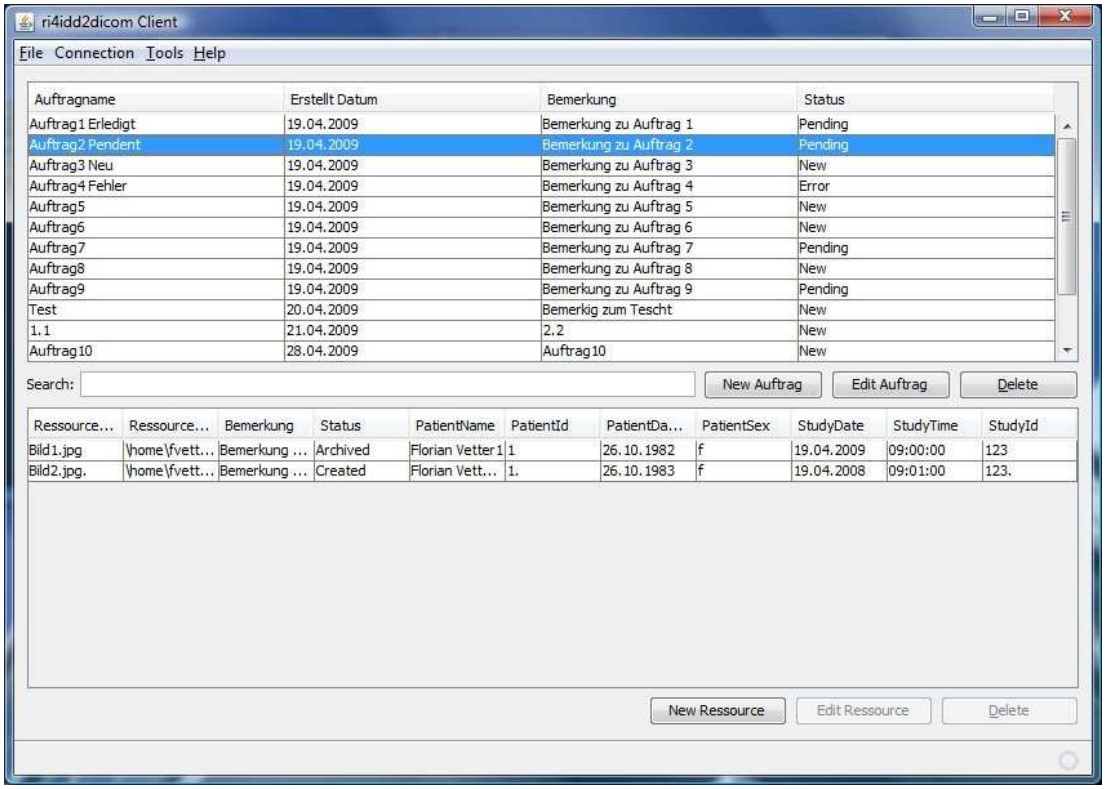


Illustration 3: Grafische Benutzeroberfläche Client

Nachdem der Client gestartet wurde wird die grafische Benutzeroberfläche geladen. Diese enthält eine Menüliste (befindet sich zu oberst), eine Statusliste (befindet sich zu unterst), sowie eine Hauptansicht (Mitte oben) und eine Detailansicht (Mitte unten). Nachfolgend eine Kurze Erläuterung zu diesen Komponenten.

3.2.1 Menüliste

Über die Menüliste können Client spezifische Aktionen ausgeführt werden. Im Menü File können Sie Aufträge erfassen, die Daten mittels *Refresh* aus dem lokalen Datenbestand aktualisieren, sowie den Client beenden.

3.2.2 Statusliste

Die Statusliste zeigt Statusinformationen während dem Betrieb dem Benutzer an.

3.2.3 Hauptansicht

Die Hauptansicht zeigt in einer Tabelle alle Aufträge an, welche vom Client verwaltet werden. Die Aufträge werden nach ihrem Erstellungsdatum automatisch sortiert.

3.2.4 Detailansicht

Die Detailansicht zeigt die entsprechenden Ressourcen an, welche in einem Auftrag zusammengefasst werden.

3.3 Benutzeraktionen

Via Menüeintrag oder Button unterhalb der jeweiligen Ansichten können durch den Benutzer unterschiedliche Aktionen ausgeführt werden.

Die Buttons werden ja nach Kontext automatisch ein- oder ausgeblendet.

Nachfolgende die Benutzeraktionen kurz erklärt:

3.3.1 Auftrag erfassen / editieren

Mittels dem Button *New Auftrag* kann ein neuer Auftrag erfasst, bzw. mit dem Button *Edit Auftrag* ein bestehender Auftrag editiert werden.

Der Auftrag ist eine Verwaltungsinstanz, welche die zu importierenden Ressourcen (Bilddateien) kapseln.

Von der Applikation wird jeweils automatisch das Erstellungsdatum gesetzt, bzw. den Auftrag mit einem entsprechenden Status (Auftragsstatus) versehen.

Mittels den unten stehenden Buttons *Save* werden die Daten gespeichert, bzw. mit dem Button *Cancel* die Änderung rückgängig gemacht.

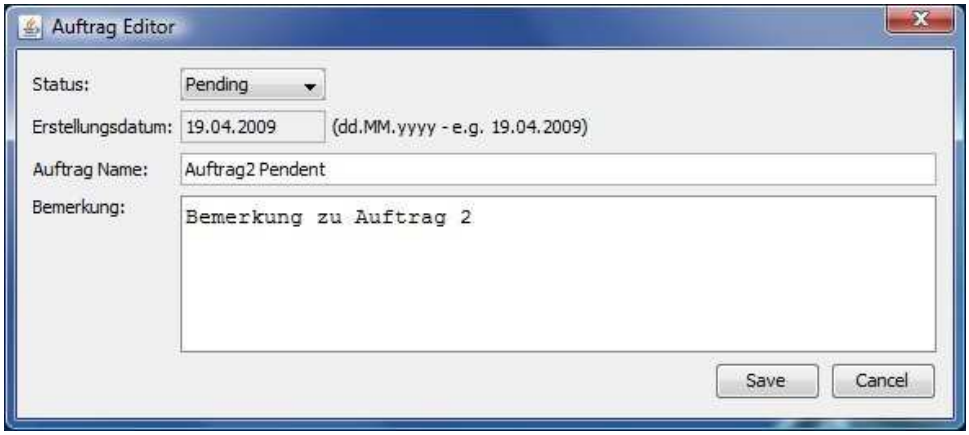


Illustration 4: Form Auftrag Editor

3.3.2 Ressource erfassen / editieren

Die Ressource entspricht den eigentlich zu importierenden Daten (z. B. eine Bilddatei). Damit eine Ressource auf dem Client erfasst werden kann, muss ein Auftrag erfasst, bzw. in der Hauptansicht selektiert sein.

Mittels dem Button *New Ressource* wird eine neue Ressource erfasst, bzw. mit dem Button *Edit Ressource* eine bestehende Ressource editiert.

Von der Applikation wird jeweils automatisch ein Status für die Ressource (Ressourcestatus) gesetzt, bzw. gewisse Daten vorgegeben.

Mittels den unten stehenden Buttons *Save* werden die Daten gespeichert, bzw. mit dem Button *Cancel* die Änderung rückgängig gemacht.

Das Formular zur Eingabe der Ressource-Informationen ist gegliedert in einen Teil Ressource und Ressource Description. Unter Ressource kann die Ressource mittels einem Selektionsfenster ausgewählt werden (Button *select Ressource* anklicken). Der Name, bzw. der Pfad dieser Ressource wird vom Client dabei automatisch übertragen und ist durch den Benutzer nicht veränderbar.

Im Teil Ressource Description werden die DICOM-Spezifischen Angaben eingetragen. Damit die Daten richtig formatiert eingegeben werden, ist für den Benutzer ein Beispieleintrag immer vorgegeben, bzw. es werden diese bei der Eingabe überprüft.

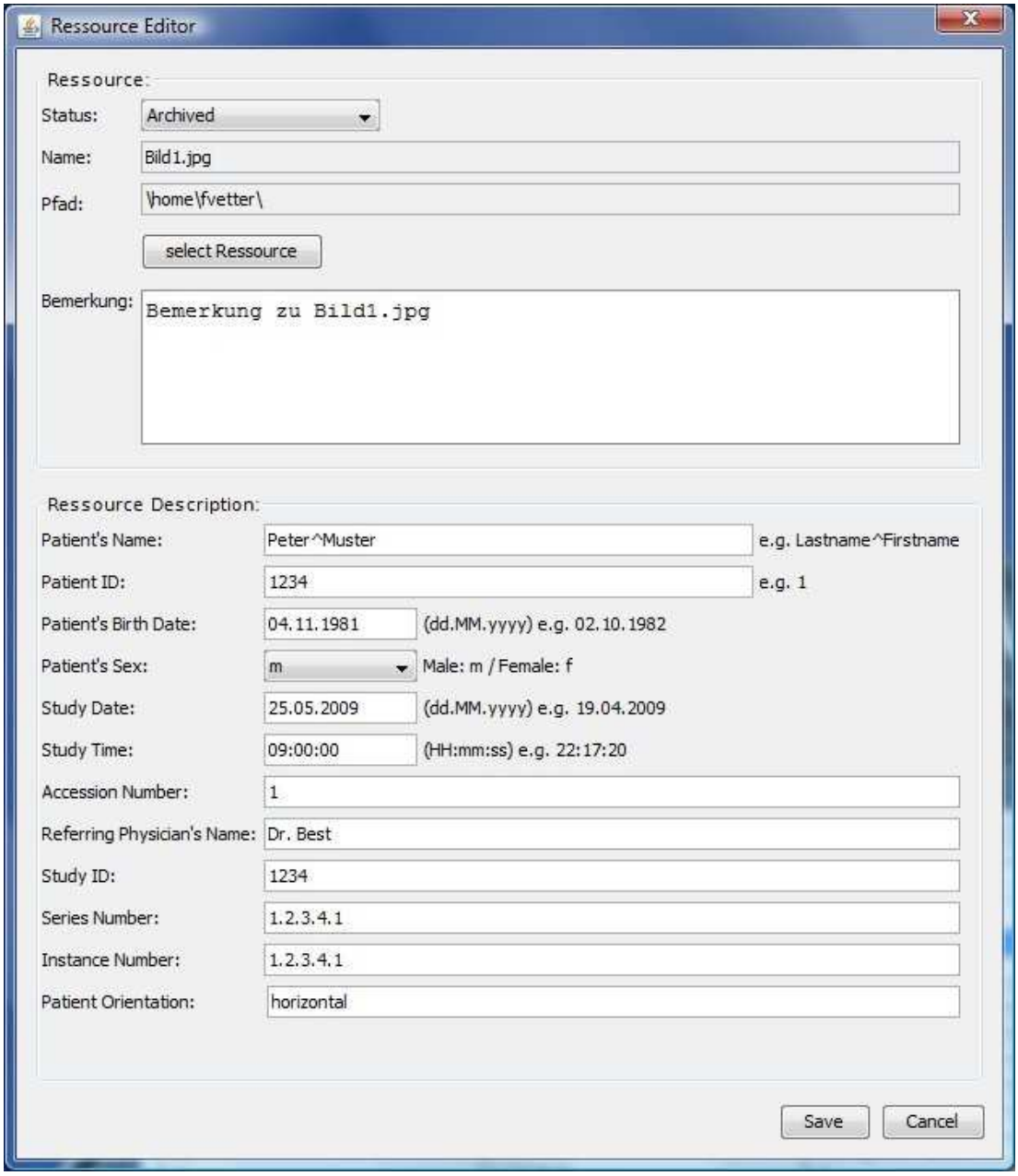


Illustration 5: Form Ressource Editor

3.3.3 Auftrag / Ressource löschen

Mittels dem Button *Delete* können die Aufträge, bzw. Ressourcen gelöscht werden. Das System fragt aber jedes mal nach, um ein versehentliches Löschen zu verhindern.

Illustration 6: Sicherheitsabfrage beim Löschen

3.3.4 Client Einstellungen ändern

Damit der Client mit dem Broker interagieren kann, müssen diverse Parameter eingegeben werden:

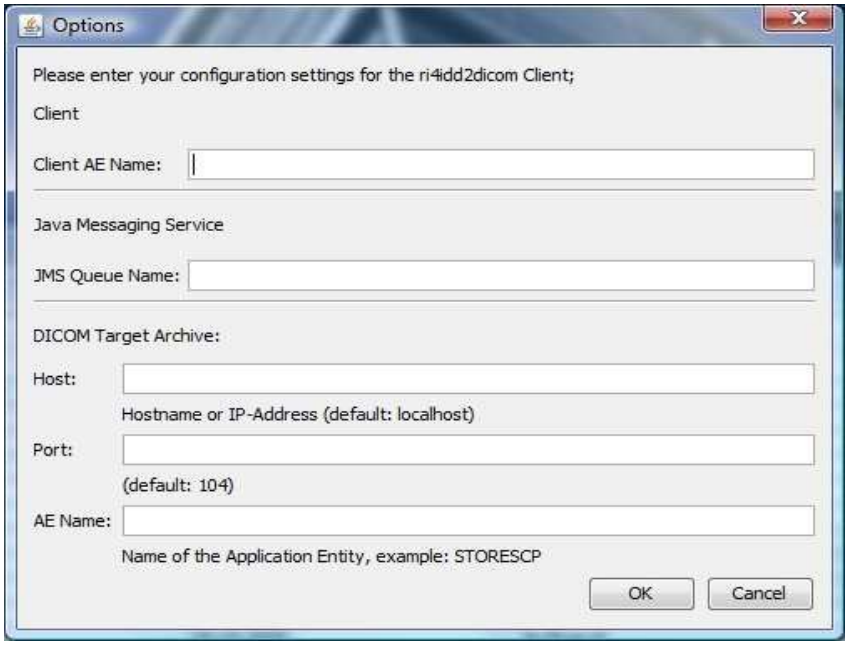


Illustration 7: Form Options

3.3.4.1 Client Einstellungsparameter

Client AE Name	DICOM Application Entity Name. Das ist ein Name, z. B. „MYCLIENT1“, welche die Applikation für die DICOM-Welt beschreibt.
JMS Queue Name	Ist ein Name, z. B. „ri4idd2dicom“, welche die Java Messaging Queue beschreibt, auf welchem die zu importierenden Daten versendet werden.
Host	Hostname oder IP-Adresse des DICOM Ziel-Archivs (DICOM-Server)
Port	Port, auf welchem der DICOM-Server seinen Dienst anbietet
AE Name	DICOM Application Entity Name, z. B. „PACSSEVER“, welche den DICOM-Server in der DICOM-Welt identifiziert.

Table 1: Client Einstellungsparameter

3.3.5 Import starten

Damit die lokal zwischengespeicherten Daten das DICOM-Zielarchiv versendet werden können, muss eine Verbindung mit dem Broker (ri4idd2dicom-broker) bestehen. Mittels Menüliste über Connection -> Connect wird der Import durch den Benutzer manuell gestartet.

3.3.6 Client beenden

Der Client kann via Menüliste über File -> Exit, bzw. durch Klicken auf den Exit-Button am Client-Fenster beendet werden.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Configurations Management Plan

21. Mai 2009

Abstract

Dieses Dokument beschreibt den Configurations Management Plan für dieses Softwareprojekt. Inhalt dabei sind alle wichtigen Informationen zum Erstellen der lauffähigen Software.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Configurations_Management_Plan_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
30.03.2009	Dokument erstellt
21.05.2009	Dokument in Version 2.0 überarbeitet
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument beschreibt den Configurations Management Plan für dieses Softwareprojekt.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Diese Dokument beschreibt welche Aktivitäten im Rahmen des Konfigurationsmanagements durchzuführen sind, wie diese auszuführen sind, wer für ihre Durchführung verantwortlich ist, wann sie durchzuführen sind und welche Mitarbeiter und Ressourcen dafür verantwortlich sind. Des weiteren finden Sie Informationen zur Installation und Konfiguration des Repositorys und der Build-Umgebung.

2 Verantwortlichkeiten

Folgende Verantwortlichkeiten können definiert werden.

Person / Rolle	Verantwortlichkeit
IT-Abteilung	Betrieb Serverinfrastruktur, z.B. SVN-Repository, Build-Server
Entwickler	Erstellen und Einchecken von Code auf SVN-Repository Deploying
Chefentwickler	Architekturdesign Bestimmen der Snap-Shots (tagging auf SVN Repository, Versionsbezeichnung innerhalb des Maven-Projekts

Table 1: Übersicht Verantwortlichkeiten

3 Verwaltung der Entwicklungsstände

Damit der Programm Code (Sourcecode) und alle wichtigen Daten für das Projekt zentral an einem Ort aufbewahrt werden kann, wurde ein Versionsverwaltungssystem, genauer ein Subversion[1] Repository, angelegt. Subversion (SVN) ist ein weit verbreitetes Versionsverwaltungssystem, welches bewährt in vielen Softwareprojekten im Einsatz steht.

3.1 SVN-Repository

Das SVN-Repository ist in der jetzigen Phase beim Studenten zu Hause auf einem virtuellen Server gehostet, kann aber auch entsprechend verschoben werden.

Das Repository wird zudem mittels einem Trac[2] für projektbezogenes Arbeiten kombiniert und ist unter folgender URL erreichbar:

https://trac.flinc.ch/svn/ri4idd2dicom/ [Abruf: 2009-05-25]

3.1.1 Allgemeine Struktur des Repositorys

Das Repository gliedert sich folgendermassen:

http://[hostname]/svn/[projectname]/

Innerhalb diesem Verzeichnis stehen 3 Unterordner (branches, tags, trunk)

Die aktuellsten Daten zum Projekt befinden sich im Ordner trunk. Die Ordner branches und tags bleiben momentan leer.

3.1.2 Beschreibung Struktur des Repositorys

Das Repository ist im Hauptordner trunk folgendermassen gegliedert:

1 http://subversion.tigris.org/
2 trac.edgewall.org

trunk

¦ - - *deploy* (enthält einen Build des ri4idd2dicom-client)

¦ ¦ - - *ri4idd2dicom-client-1.0* (Client jar mit db)

¦ - - *doc* (enthält alle wichtigen Daten für die Dokumentation)
¦ ¦ - - *Bilder* (enthält alle in dieser Arbeit verwendeten Bilder)

¦ ¦ - - *Diagramme* (enthält alle in dieser Arbeit erstellten Diagramme)

¦ ¦ - - *DICOM* (enthält alle Daten zu DICOM)

¦ ¦ - - *Dokumente*

¦ ¦ ¦ - - *final* (enthält die fertige Dokumentation)

¦ ¦ - - *Projektmanagement* (enthält alle Projektplan und Zeiterfassung)

¦ ¦ - - *Protokolle* (enthält alle Protokolle und Traktandenlisten nach Wochen)

¦ ¦ - - *Unterlagen* (enthält Dokumente mit relevanten Informationen)

¦ ¦ - - *Vorlagen* (enthält alle Dokumentvorlagen)

¦ - - *lib* (spezielle Bibliotheken)

¦ - - *prototype* (enthält den Prototpen)

¦ - - *ri4idd2dicom* (enthält das Softwareprojekt)

¦ ¦ - - *ri4idd2dicom-broker* (Softwareprojekt des Broker)

¦ ¦ - - *ri4idd2dicom-client* (Softwareprojekt des Clients)

¦ - - *scripts* (enthält die Scripts zum Erstellen der Datenbank)

3.1.3 Arbeiten mit dem Repository

Alles was man braucht um mit dem Repository zu arbeiten ist ein SVN-Client. Ein bewährter Client ist beispielsweise der TortoiseSVN-Client.

Für Informationen zur Installation des SVN-Repositorys wird auf den Abschnitt 4 Installation dieses Dokumentes verwiesen.

3.1.3.1 Sourcecode auschecken (TortoiseSVN)

Mittels einem SVN-Client kann der Sourcecode ausgechecked werden. Der TortoiseSVN-Client bietet eine einfache Integration über das Kontextmenü im Windows Explorer an.

Den aktuellsten Stand der Entwicklung auszuchecken ist sehr einfach:

1. Neues Verzeichnis erstellen, wo der Quellcode abgelegt werden soll. Beispielsweise C:\SVN_Projekte\ri4idd2dicom.
2. Mit rechter Maustaste auf den Ordner ri4idd2dicom klicken und die Option SVN Checkout auswählen.
3. Wenn der Client nach der URL des Repositorys fragt, bitte die URL aus dem obigen Abschnitt eingeben.
4. OK klicken
5. Je nachdem verlangt Tortoise eine Authentifizierung mit Benutzername und

Passwort. Verwenden Sie bitte ihr zugewiesenes Passwort.

Das Sourceverzeichnis enthält eine menge an Dateien. Sind Sie sich darum bewusst, dass der Initial-Import in ihr lokales Repository eine gewisse Zeit dauert. bis der ganze Verzeichnisbaum vollständig heruntergeladen wurde.

3.1.3.2 Sourcecode auschecken (Linux SVN Client)

Software für Subversion installieren:

```
$ sudo apt-get install subversion libsvn-java
```

Repositoryverzeichnis lokal anlegen: /home/myusername/sa2/repository

```
$ svn checkout https://trac.flinc.ch/svn/ri4idd2dicom/trunk

$ svn import -m "renamed the trunk folder"
/home/myusername/sa2/repository/ri4idd2dicom https://trac.flinc.ch/svn/ri4idd2dicom/
```

Beim Einsatz von Eclipse sind folgende Anpassungen in Eclipse noch vorzunehmen:

Datei eclipse.ini im Eclipse Ordner editieren und mit folgender Zeile ergänzen:

```
$ sudo nano /opt/eclipse/eclipse.ini

-Djava.library.path=/usr/lib/jni
```

3.1.4 Client-Befehle

Um das Repository aktualisieren zu können muss zuerst ins lokale Repository-Verzeichnis gewechselt. Folgende Befehle sind dabei relevant:

3.1.4.1 Update

```
$ svn update
```

3.1.4.2 Commit

```
$ svn commit
```

3.1.4.3 Hinzufügen und löschen von Dateien aus dem Projekt

```
$ svn add <Dateiname>
```

Files zum löschen „vormerken“:

```
$ svn delete <Dateiname>

$ svn move <Quelle> <Ziel>
```

Die Änderungen werden beim nächsten Commit in das Repository übertragen. Für weitere Parameter zur Nutzung des SVN-Clients unter Linux siehe die Hilfe via Konsole.

3.1.5 Regeln im Umgang mit dem Repository

Bevor Sie als Entwickler neue Daten bei sich lokal erstellen, vergewissern Sie sich mittels einem SVN-Update, dass Sie lokal den aktuellsten Stand wie auf dem SVN-Repository

besitzen.

Bevor Sie als Entwickler neue Daten ins Repository einchecken, stellen Sie sicher, dass diese Daten einen guten Stand aufweisen (d.h. funktionierenden Code), damit andere Personen damit arbeiten können.

Die aktuellen Daten befinden sich immer innerhalb des „Trunk“-Unterverzeichnisses des SVN-Repositorys.

3.1.6 Finden der Entwicklungszustände

Jede eingecheckte Datei wird durch das SVN-Repository verwaltet. Bemerkt man beispielsweise, dass man auf einen früheren Stand zurückgreifen muss, so kann via SVN-Checkout die gewünschte Revision angegeben werden.

Diese Lösung ist sehr elegant und erlaubt einem somit festzustellen, was alles geändert hat.

TortoiseSVN bietet mit dem Repository Browser ein Werkzeug an, um die Entwicklungszustände der eingecheckten Dateien zu durchsuchen.

4 Installation

4.1 Repository

Für diese Studienarbeit wurde eine eigenständiges Repository auf einem virtuellen Server eingerichtet. SVN[3] als Versionsverwaltungssystem in Kombination mit der Softwareprojektverwaltungslösung Trac[4] stellen bewährte Hilfsmittel für eine moderne Softwareentwicklung zur Verfügung.

4.1.1 Trac

Für Informationen zu Trac kann auf die offizielle Webseite des Projektes verwiesen werden:

4.1.1.1 Installationsanleitungen

Auf der Seite von Trac existieren diverse Installationsanleitungen. Folgende Links waren für die Installation hilfreich:

http://trac.edgewall.org/wiki/TracInstall [Abruf: 2009-05-25]

http://trac.edgewall.org/wiki/DatabaseBackend#Postgresql [Abruf: 2009-05-25]

http://trac.edgewall.org/wiki/TracEnvironment#DatabaseConnectionString [Abruf: 2009-05-25]

4.1.1.2 PostgreSQL Konfiguration

PostgreSQL wurde mit der Grundinstallation des Ubuntu Servers bereits mitinstalliert. Als nächstes müssen wir die DB entsprechend konfigurieren. Folgende Benutzer mit entsprechenden Passwörter sind vorgesehen:

postgres user:	postgres	pw:	postgres
database user:	tracuser	pw:	tracuser
database name:	trac		

- PostgreSQL Konfigurationsdatei editieren:

```
$ sudo nano /etc/postgresql/8.3/main/postgresgl.conf

# - Connection Settings -listen addresses = '*'
# - Security and Authentication -

ssl_ciphers = 'ALL:!ADH:!LOW:!EXP:!MD5:@STRENGTH'  # allowed SSL ciphers
password_encryption = on
```

- Einstellen der Zugriffsbeschränkung auf IP-Basis:

```
$ sudo nano /etc/postgresql/8.3/main/pg_hba.conf

# IPv4 local connections:
host    all         all         192.168.10.0/24       md5
```

- Ändern des Passwortes des Postgres Benutzers auf Seiten der Datenbank

3 http://subversion.tigris.org/
4 http://trac.edgewall.org/

```
$ sudo -u postgres psql postgres

postgres=# ALTER USER postgres WITH ENCRYPTED PASSWORD 'postgres';
postgres=# \q
```

- Passwort von Benutzer ändern (Passwort: „postgres“)

```
$ sudo passwd postgres
```

- Datenbank für Trac erstellen;

```
$ sudo -u postgres createdb trac
```

- Benutzer wechseln:

```
$ su postgres

$ createuser -U postgres -E -P tracuser

$ sudo /etc/init.d/postgresql-8.3 restart

$ sudo apt-get install sqlite3
```

4.1.1.3 Installation

http://trac.edgewall.org/wiki/0.10.4/TracOnUbuntuHardy [Abruf: 2009-05-25]

Installation Apache Webserver:

```
$ sudo apt-get install apache2
```

Installation Subversion:

```
$ sudo apt-get install subversion libapache2-svn

$ sudo nano /etc/apache2/mods-enabled/dav_svn.conf



<Location /svn/>
  DAV svn
  SVNParentPath /var/lib/svn
  #Provides dirctory listing of repositories
  AuthType Basic
  AuthName "Subversion Repository"
  AuthUserFile /etc/apache2/dav_svn.passwd
  AuthzSVNAccessFile /etc/apache2/dav_svn.authz
  Require valid-user
 </Location>
```

```
 $ sudo htpasswd -cm /etc/apache2/dav_svn.passwd username1

 $ sudo htpasswd /etc/apache2/dav_svn.passwd username2

New password: my_password
Re-type new password: my_password
```

```
 $ sudo nano /etc/apache2/dav_svn.authz

 [ri4idd2dicom:/]
 username1 = rw
 username2 = r
```

- Appache Webserver neu starten

```
 $ sudo /etc/init.d/apache2 restart
```

- Als nächstes muss das Subversion-Verzeichnis erstellt werden und danach wird ein erstes Projekt (ri4idd2dicom) angelegt:

```
 $ sudo mkdir /var/lib/svn

 $ sudo svnadmin create /var/lib/svn/ri4idd2dicom

 $ sudo mkdir -p /tmp/svn/template/{branches,tags,trunks}

 $ sudo chown -R www-data:www-data /var/lib/svn

 $ sudo svn import /tmp/svn/template http://localhost/svn/ri4idd2dicom -m "initial
import" --username fvetter
```

- Testen, d.h. Verbinden mit Webbrowser:
  http://192.168.10.52/svn/ri4idd2dicom

4.1.1.4 Trac

Siehe: http://trac.edgewall.org/wiki/0.11/TracOnUbuntu [Abruf: 2009-05-25]

- Notwendige Komponente installieren:

```
$ sudo apt-get install  libapache2-mod-python python-setuptools python-subversion
```

- Trac Installation beginnen:

```
$ sudo easy_install Trac
```

- Trac Verzeichnis erstellen und Benutzerrechte anpassen:

```
$ sudo mkdir /var/lib/trac
$ sudo chown -R www-data:www-data /var/lib/trac
```

- Folgender Inhalt in die Datei trac hinzufügen:

```
$ sudo nano /etc/apache2/sites-available/trac

# Trac Configuration
<VirtualHost *:80>
 ServerName trac.flinc.ch
 Redirect / https://trac.flinc.ch/projects/
</VirtualHost>
<VirtualHost *:443>
       ServerAdmin webmaster@flinc.ch
       ServerName trac.flinc.ch
       DocumentRoot /var/www
       ErrorLog /var/log/apache2/error.trac.log
       CustomLog /var/log/apache2/access.trac.log combined
       SSLEngine on
       SSLCertificateFile /etc/apache2/ssl/trac.crt
       SSLCertificateKeyFile /etc/apache2/ssl/trac.key

       <Location /projects>
                SetHandler mod_python
                PythonInterpreter main_interpreter
                PythonHandler trac.web.modpython_frontend
                PythonOption TracEnvParentDir /var/lib/trac
                PythonOption TracUriRoot /projects
                PythonOption PYTHON_EGG_CACHE /tmp
```

```
        </Location>

        # use the following for one authorization for all projects
        # (names containing "-" are not detected):
        <LocationMatch "/projects/[[:alnum:]]+/login">
            AuthType Basic
            AuthName "trac"
              # Use the SVN password file.
            AuthUserFile /etc/apache2/dav_svn.passwd
            Require valid-user
        </LocationMatch>
</VirtualHost>
```

- SSL Verschlüsselung einrichten:

```
$ cd

$ openssl genrsa -aes256 -out trac.key 2048

$ openssl req -new -x509 -days 365 -key trac.key -out trac.crt
```

- Zertifikatsangaben ausfüllen:

```
Country Name (2 letter code) [AU]:CH

State or Province Name (full name) [Some-State]:ZH

Locality Name (eg, city) []:Zuerich

Organization Name (eg, company) [Internet Widgits Pty Ltd]:Home

Organizational Unit Name (eg, section):

Common Name (eg, YOUR name) []:Administrator

Email Address []:administrator@flinc.ch
```

- Dateirechte für Zertifikat anpassen und in entsprechenden Ordner verschieben:

```
$ chmod 400 trac.key trac.key.org

$ chown root:root trac.key trac.key.org

$ sudo mv trac.crt /etc/apache2/ssl/

$ sudo mv trac.key /etc/apache2/ssl/
```

- Folgende Datei unter <VirtualHost *:433> anpassen:

```
$ sudo nano /etc/apache2/sites-available/trac

SSLCertificateFile /etc/apache2/ssl/trac.crt
SSLCertificateKeyFile /etc/apache2/ssl/trac.key
```

- Appache Seiten neu laden:

```
$ sudo a2dissite default

$ sudo a2ensite trac

$ sudo /etc/init.d/apache2 reload

$ sudo mkdir /var/lib/trac

$ sudo trac-admin /var/lib/trac/YourProjectNameHere initenv

$ sudo chown -R www-data /var/lib/trac
```

- Testen ob SSL-Verschlüsselung aktiv ist beim Aufruf der Trac-Seite: http://trac.flinc.ch/projects/ri4idd2dicom

4.1.1.5 Konfiguration

Für das Projekt sollte ein kurzer Projektname ausgewählt werden, z. B.

[Projektname]: ri4idd2dicom

Pfad auf das Trac-Verzeichnis: /var/lib/trac/[Projektname]

Pfad auf das SVN-Verzeichnis: /var/lib/svn/[Projektname]

- Passwortfile für SVN-Repository erstellen:

```
$ ntpasswd -c /etc/apache2/dav_svn.passwd [username]

$ sudo find /var/lib/svn/[projectname] -type f -exec chmod 660 {} \;

$ sudo find /var/lib/svn/[projectname] -type d -exec chmod 2770 {} \;

$ sudo svn import /tmp/[projectname] file:///var/lib/svn/[projectname] -m "initial
import"

$ find /var/lib/trac -type f
```

- Trac Projektumgebung einrichten:

```
$ sudo trac-admin /var/lib/trac/[projectname] initenv
```

- Datenbank-Verbindungs String:

```
postgres://tracuser:tracuser@localhost/trac
```

Bei einem Fehler muss evtl. Folgendes Paket nachinstalliert werden:

```
$ sudo apt-get install python2.5-psycopg
```

4.2 Build Umgebung

Die Build-Umgebung soll sicherstellen, dass automatische Builds nach jedem Einchecken in die Mainline des Projektes erstellt werden. Dabei wird automatisch die Funktionalität des neuen Codes ausgetestet und falls o.k. ein lauffähiger Build erzeugt.

4.2.1 Ziele

- Ablaufen von System und Integrationstests
- Erstellen eines täglichen Builds (nightly build)

4.2.2 CruiseControl

CruiseControl[5] ist ein java-basiertes Computerprogramm, welche in der Softwareentwicklung zum automatischen Generiern von Builds verwendet wird.

System enthält eine Weboberfläche zum anzeigen des aktuellen Systemzustandes des Systems, sowie den vorhergegangenen (History).

Für Installationsanleitung siehe:

http://cruisecontrol.sourceforge.net/gettingstartedbindist.html [Abruf: 2009-05-25]

http://www.testearly.com/2007/03/24/configuring-cruisecontrol-in-linux/ [Abruf: 2009-05-25]

4.2.3 Installation

- Virtuellen Server starten
- SSH Verbindung mit Server herstellen:

```
$ ssn H92.168.168.12
```

- Java6 JDK installieren:

```
$ sudo apt-get install sun-java6-jdk
```

- Variable JAVA_HOME setzen und folgende Zeile hinzufügen:

```
$ sudo nano /etc/environment

JAVA_HOME="/usr/lib/jvm/java-6-sun/jre"
```

- CruiseControl in ein temporäres Verzeichnis downloaden:
  siehe Webseite: http://cruisecontrol.sourceforge.net/download.html [Abruf: 2009-05-25]
  -> aktuellste Version 2.8.2 (cruisecontrol-bin-2.8.2.zip)

```
$ cd
$ wget http://switch.dl.sourceforge.net/sourceforge/cruisecontrol/cruisecontrol-bin-2.8.2.zip
```

- Programm unzip installieren um Datei zu entpacken:

5 http://cruisecontrol.sourceforge.net/

```
$ sudo apt-get install unzip
```

- Zip-Archiv ins Zielverzeichnis extrahieren, wo man CruiseControl installieren möchte:

```
$ unzip cruisecontrol-bin-2.8.2.zip

$ sudo mv cruisecontrol-bin-2.8.2 /opt/
```

- Benutzer und Benutzergruppe erstellen, sowie sicherstellen, dass die CruiseControl Dateien die neuen Benutzer als Besitzer haben
  z.B. als Root-Benutzer folgender Benutzer (cimaster) und Gruppe (cimasters) erstellen:

```
$ sudo groupadd cimasters
$ sudo useradd cimaster
$ sudo usermod -a -G cimasters cimaster
```

- Überprüfen, ob Benuterz und Gruppen angelegt wurden:

```
cat /etc/group | grep cimasters
```

- Passwort von Benutzer cimaster ändern: (pw=cimaster)

```
$ sudo passwd cimaster
```

- Als nächstes alle Besitzer des CruiseControl Ordners ändern:

```
$ sudo chown -R cimaster:cimasters /opt/cruisecontrol-bin-2.8.2/
```

- Datei erstellen um CruiseControl als Service zu starten.
  Dieser Code basiert auf dem CruiseControl Service Code[6] vom CruiseControl Wiki.
- Konfigurationsdatei erstellen und Inhalt aus Wiki in Datei cruisecontrol auf Server hinein kopieren:

```
$ sudo nano /etc/init.d/cruisecontrol
```

- Dateirechte anpassen:

```
$ sudo chmod 755 cruisecontrol
```

Für weitere Informationen siehe folgende Links:

http://cruisecontrol.sourceforge.net/overview.html [Abruf: 2009-05-25]

http://confluence.public.thoughtworks.org/display/CC/Home [Abruf: 2009-05-25]

6 http://confluence.public.thoughtworks.org/display/CC/RunningCruiseControlFromUnixInit

5 Konfigurationen

Nachfolgend eine Übersicht der im Projekt vorgenommenen Konfigurationen.

Das Projekt wurde mit dem Kurznamen ri4idd2dicom abgekürzt.

5.1 SVN-Repository

Das Repository befindet sich auf einem Speziellen Server, welcher mehrere Repositorys verwaltet.

http://[hostname]/svn/[projectname]/

Für jedes Projekt werden 3 Unterordner (branches, tags, trunk) definiert.

http://[hostname]/svn/[projectname]/{ branches | tags | trunk}

5.2 Maven Projektkonfiguration

Für das Projekt ist ein entsprechendes Maven Projekt erstellt worden, welche seine Konfigurationen in einem Konfigurationsfile (pom.xml) verwaltet. Im Hauptprojekt werden auf die Subprojekte (Module), welche ebenfalls als Mavenprojekte definierte wurden, entsprechend verwiesen.

Folgende Definitionen sind im Maven Konfigurationsfile (pom.xml) definiert

Artifact

Group id: ch.hsr.app

Artifact id: ri4idd2dicom

Version: 1.0-SNAPSHOT

Packing: pom

Project

Name: ri4idd2dicom

Description: Softwareprojekt für Studienarbeit FS 2009 an der HSR

Modules

ri4idd2dicom-broker

ri4idd2dicom-client

6 Build-Vorgang

Damit der Client Code auf eine Arbeitsstation des Benutzers deployed werden kann, muss dieser entsprechend kompiliert und in einem JAR-Archiv verpackt werden.

Durch den Einsatz des Build-Management-Tools Maven[7] kann dies sehr einfach und elegant durchgeführt werden, da Maven ein Dependency Management implementiert. Werden in einem Modul entsprechende JAR-Files verwendet, sind diese innerhalb der Maven-Konfigurationsdatei (pom.xml) definiert. Wird später der fertige Client im Build-Prozess

7 http://maven.apache.org/

erstellt, so werden die benötigten JAR-Files automatisch durch Maven bereitgestellt.

Bevor man jedoch mit dem Build beginnen kann muss sichergestellt werden, dass auf der Build-Maschine Maven installiert ist.

Unter Linux kann Maven mittels folgendem Befehl installiert werden:

```
$ sudo apt-get install maven2
```

Zum Überprüfen ob Maven richtig installiert wurde, kann in der Konsole durch Eingabe des Befehls mvn --version überprüft werden:

```
$ mvn --version
Maven version: 2.0.9
Java version: 1.6.0_10
OS name: "linux" version: "2.6.27-11-generic" arch: "amd64" Family: "unix"
```

Innerhalb der Maven Projektdatei (pom.xml) des Clients wurden entsprechende Konfigurationen vorgenommen, um den Buildvorgang durch Maven automatisch durchführen zu können (siehe unten stehender XML Auszug).

```
<plugin>
   <artifactId>maven-assembly-plugin</artifactId>
   <configuration>
      <descriptorRefs>
         <descriptorRef>jar-with-dependencies</descriptorRef>
      </descriptorRefs>
      <archive>
         <manifest>
            <mainClass>ch.hsr.ri4idd2dicom.gui.Ri4idd2dicomApp</mainClass>
         </manifest>
      </archive>
   </configuration>
      <executions>
         <execution>
            <id>make-assembly</id>
            <!-- this is used for inheritance merges -->
            <phase>package</phase>
            <!-- append to the packaging phase. -->
            <goals>
               <goal>attached</goal>
               <!-- gals == mojos -->
            </goals>
         </execution>
      </executions>
</plugin>
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-jar-plugin</artifactId>
   <configuration>
      <archive>
         <manifest>
            <addClasspath>true</addClasspath>
            <mainClass>ch/hsr/ri4idd2dicom/gui/Ri4idd2dicomApp</mainClass>
         </manifest>
      </archive>
   </configuration>
</plugin>
<plugin>
   <artifactId>maven-compiler-plugin</artifactId>
   <version>2.0.2</version>
   <configuration>
      <source>1.6</source>
      <target>1.6</target>
   </configuration>
</plugin>
```

Table 2: Auszug aus der pom.xml Datei des Clients

Aus dem pom.xml Auszug wird der Einsatz diverser maven plugins sichtbar. Es sind dies:

Maven-Plugin	Beschreibung
maven-assembly-plugin	Wird verwendet zum Definieren der zu erstellenden Jar-Datei (wichtig ist der Eintrag <mainClass>, damit die Applikation später gestartet werden kann). Weiterführende Informationen zum Einsatz des Maven Jar Plugins finden Sie unter folgender URLs: http://maven.apache.org/plugins/maven-jar-plugin/usage.html http://maven.apache.org/shared/maven-archiver/examples/classpath.html
maven-jar-plugins[8]	Mittels dem maven-assembly-plugin wird der Prozess zum Assemblieren, d.h. Zusammenstellen des kompletten Jar-Archives sichergestellt. Während diesem Prozess werden die im Projekt als abhängig definierten Jar-Librarys aus dem lokalen Maven-Repository in das fertige Jar-Archiv importiert, damit der Client das Projekt, mit allen notwendigen Jar-Komponenten, ausführen kann.
maven-compiler-plugin	Mittels dem maven-compiler-plugin wird die Java Kompatibilität auf die Versoin 1.6 festgelegt, welches für die Ausführung des Clients vorgegeben ist.

Table 3: verwendete Maven-Plugins im Client

6.1 Buildvorgang Step-by-Step

Für den Buildvorgang kann eine dedizierte Maschine eingesetzt werden (Build-Server), oder auch die Arbeitsstation des Entwicklers, welche über eine Maven-Installation verfügt:

- Öffnen Sie auf der Build-Maschine ein Konsolenfenster
- Wechseln Sie in das Verzeichnis, wo sich das Client-Projekt befindet

```
$ cd sa2/repository/trunk/ri4idd2dicom/ri4idd2dicom-client/
```

- Als nächstes wird mittels dem Maven-Befehl assembly:assembly der Assemblierungs-Prozess angestossen.

```
$ mvn assembly:assembly
```

- Maven führt nun automatisch die verschiedenen Prozesse (wie Build, Tests und Assemblieren) aus.

```
[INFO] Scanning for projects...
[INFO] Searching repository for plugin with prefix: 'assembly'.
[INFO] ------------------------------------------------------------------------
[INFO] Building ri4idd2dicom-client
[INFO]    task-segment: [assembly:assembly] (aggregator-style)
[INFO] ------------------------------------------------------------------------
[INFO] Preparing assembly:assembly
[INFO] ------------------------------------------------------------------------
[INFO] Building ri4idd2dicom-client
[INFO] ------------------------------------------------------------------------
[INFO] [resources:resources]
```

8 uhttp://maven.apache.org/plugins/maven-jar-plugin/

```
[INFO] Using default encoding to copy filtered resources.
[INFO] [compiler:compile]
[INFO] Nothing to compile - all classes are up to date
[INFO] [resources:testResources]
[INFO] Using default encoding to copy filtered resources.
[INFO] [compiler:testCompile]
[INFO] Nothing to compile - all classes are up to date
[INFO] [surefire:test]
[INFO] Surefire report directory:
/home/fvetter/sa2/repository/trunk/ri4idd2dicom/ri4idd2dicom-client/target/surefire-
reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running ch.hsr.ri4idd2dicom.manager.RessourceTest
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.049 sec
Running ch.hsr.ri4idd2dicom.DicomRessourceTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.067 sec

Results :

Tests run: 3, Failures: 0, Errors: 0, Skipped: 0

[INFO] [jar:jar]
[INFO] [assembly:attached {execution: make-assembly}]
[INFO] Processing DependencySet (output=)
[INFO] Building jar: /home/fvetter/sa2/repository/trunk/ri4idd2dicom/ri4idd2dicom-
client/target/ri4idd2dicom-client-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO] [assembly:assembly]
[INFO] Processing DependencySet (output=)
[INFO] Building jar: /home/fvetter/sa2/repository/trunk/ri4idd2dicom/ri4idd2dicom-
client/target/ri4idd2dicom-client-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 11 seconds
[INFO] Finished at: Wed May 13 12:28:13 CEST 2009
[INFO] Final Memory: 25M/353M
[INFO] ------------------------------------------------------------------------
```

- Ist dieser Prozess abgelaufen, dann liegt im target Verzeichnis innerhalb des Client-Projekts eine fertige Jar-Datei vor, welche mit allen Abhängigkeiten (verlinkte Jar-Dateien) in einem Snapshot gepackt ist.
- wechseln in den target Ordner

```
$ cd /target
```

- kopieren Sie die erstellte Datei in ein separates Deploy-Verzeichnis

```
$ cp ri4idd2dicom-client-1.0-SNAPSHOT-jar-with-dependencies.jar
/home/myusername/Desktop/deploy/
```

- Als nächstes muss die Datenbank noch in das Deploy-Verzeichnis kopiert werde, damit der Client richtig gestartet werden kann.
- Erstellen Sie im Deploy-Verzeichnis einen Ordner mit Name db:

```
$ mkdir db
```

- Kopieren Sie die Datenbankfiles aus dem Repository in den oben erstellten Datenbank ordner im Deploy-Verzeichnis:

```
$ cp trunk/ri4idd2dicom/ri4idd2dicom-client/db/ri4idd2dicomdb.*
/home/myusername/Desktop/deploy/db/
```

- Wechseln Sie nun in das Deploy Verzeichnis

```
$ cd /home/myusername/Desktop/deploy/
```

- Mittels nachfolgendem Befehl kann der Client gestartet werden

```
$ java -jar  ri4idd2dicom-client-1.0-SNAPSHOT-jar-with-dependencies.jar
```

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Systemtests

27. Mai 2009

Abstract

Dieses Dokument beschreibt die durchgeführten Systemtests in dieser Studienarbeit, welche sich auf die Funktionalitäten beziehen, welche im Dokument Anforderungsspezifikation festgehalten sind.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Systemtests_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
30.03.2009	Dokument erstellt
12.05.2009	Dokument in Version 2 überarbeitet
14.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument beschreibt die durchgeführten Systemtests in dieser Studienarbeit, welche sich auf die Funktionalitäten beziehen, welche im Dokument Anforderungsspezifikation festgehalten sind.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Inhaltlich werden die in diesem Dokument aufgelisteten Komponenten nachfolgenden Systemtests unterzogen.

2 Vorgehensweise

Die Systemtests werden jeweils nach folgendem Vorgehen durchgeführt:

1. Testszenario (Was wird getestet, welche Anforderung)
2. Erwartung (Kriterien für diesen Test, was wird erwartet)
3. Resultat (Beschreibung der Testdurchführung mit Resultat)

2.1 Bemerkungen

Da die Komponenten in der Durchführungszeit dieser Studienarbeit funktional nicht fertig erstellt werden konnten, sind die Systemtests nicht so ausführlich ausgefallen.

Die Möglichkeit der Integrations-, bzw. Systemtests (Unit-Testing) wurden zu wenig genutzt, da zuerst die grundsätzlichen Funktionalität sichergestellt werden mussten. Wäre der Client fertig geworden, wäre das Ziel gewesen diesen mit entsprechenden Systemtests, bzw. Integrationstest auszustatten.

3 Systemtests

Die nachfolgenden Systemtests wurden manuell für die entsprechenden Komponenten durchgeführt:

3.1 Client

3.1.1 Ressource in DICOM-Objekt umwandeln

Testszenario:	Umwandlung einer Ressource (Jpeg-Datei) in ein DICOM-Objekt Anforderungen: Ressource (Jpeg-Datei), inkl. einer Beschreibung zur Ressource muss vorhanden sein. Es wird die Funktionalität aus dem dcm4che2 Toolkit verwendet
Erwartung:	Eine einfache Ressource wird mittels der DICOM Konvertierungsfunktionalität in ein DICOM-Objekt umgewandelt, welches danach die DICOM-Konformität erfüllt.
Resultat:	Resultat: Die Ressource liegt in einem DICOM-Objekt vor, Test ist daher erfolgreich abgelaufen.

Table 1: Systemtest Ressource in DICOM-Objekt umwandeln

3.1.2 DICOM Ressource senden

Testszenario:	Senden der Ressource (DICOM-Objekt) an den DICOM-Server Anforderung: Ressource (DICOM-Objekt) muss umgewandelt sein, Verbindung zu DICOM-Server vorhanden
Erwartung:	Das DICOM-Objekt (Ressource) wird an den DICOM-Server gesendet, welche die Ressource entgegen nimmt und abspeichert.
Resultat:	DICOM-Objekt wird mittels der DICOM-Sendekomponente an den DICOM-Server gesendet. Resultat: Test ist erfolgreich, die Ressource wurde vom DICOM-Server angenommen und abgespeichert.

Table 2: Systemtest DICOM Ressource senden

3.1.3 DICOM Storage Commitment

Testszenario:	Ausführen eines Storage Commitment für eine Ressource, welche an den DICOM-Server gesendet wurde. Anforderungen: Ressource wurde in ein DICOM-Objekt umgewandelt, Ressource (DICOM-Objekt) wurde an den DICOM-Server gesendet.
Erwartung:	Bestätigtes Storage Commitment für die Ressource durch den DICOM-Server.
Resultat:	In der DICOM-Sendekomponente des Clients wird für die gesendete Ressource ein Storage Commitment verlangt. Resultat: Storage Commitment bricht mit einem Fehler A801H ab. Dieser Fehlercode bedeutet „Refused Move destination unknown“. Die Funktionalität des Storage Commitments muss nochmals überprüft werden, d.h. Test ist nicht erfolgreich.

Table 3: Systemtest DICOM Storage Commitment

3.1.4 Build

Testszenario:	Erstellen eines Build für den Client, welche danach für die Installation des Client genutzt werden kann. Anforderungen: Client ist als maven-Projekt definiert und die entsprechenden Build-Konfigurationen definiert.
Erwartung:	Der Buildvorgang wird durch maven sichergestellt. Nach dem Buildvorgang liegt eine einzige Jar-Datei vor, welche den Client inkl. seinen notwendigen Jars beinhaltet.
Resultat:	Der Test wird nach der Beschreibung des Build-Vorgangs durchgeführt, welcher im Dokument *Configurations Management Plan* beschrieben wird. Resultat: Build hat ohne Probleme funktioniert, Test wurde erfüllt.

Table 4: Systemtest Build

3.1.5 Client starten und beenden

Testszenario:	Client starten und beenden Anforderungen: es liegt ein Build des Clients vor und dieser wurde gemäss dem Dokument *Benutzerdokumentation* in Betrieb genommen.
Erwartung:	Dieser Systemtest soll das Zusammenspielen der eingesetzten Komponenten (GUI, Datenbank) überprüfen. Es wird erwartet, dass beim Starten des Clients die Daten aus der Datenbank geladen und diese im GUI angezeigt wird. Beim beenden wird der Client geschlossen und dabei die Daten in die Datenbank geschrieben.
Resultat:	Client starten, wenn GUI angezeigt wird Client wieder beendet. Resultat: Das Starten und Beenden des Client (Daten werden entsprechend aus der Datenbank geladen und wieder abgespeichert) funktioniert, d.h. der Test wurde erfüllt.

Table 5: Systemtest Applikation starten und beenden

3.1.6 Datenspeicherung (Datenbank)

Testszenario:	Austesten der Datenspeicherung (Persistieren) der Daten in die Datenbank über die Client-Applikation (GUI). Anforderung: Client besitzt einen Persistenz Provider, welche im Hintergrund die Verwaltung der Daten auf die Datenbank sicherstellt
Erwartung:	Der Client realisiert die notwendige Funktionalität zum Erzeugen/Bearbeiten und Löschen der Objekte, welche nach Beenden der Benutzeraktion auf die Datenbank geschrieben werden.
Resultat:	Client starten, Daten werden aus der Datenbank geladen und im Client-GUI angezeigt. Via Benutzereingabe wird einen Auftrag erfasst. Als nächstes wird zu einem Auftrag eine Ressource erstellt und die entsprechenden Daten eingegeben, bzw. diese danach abgespeichert. Danach wird der Client wieder beendet und nochmals gestartet und die vorhin eingegebenen Daten auf Vollständigkeit überprüft. Es wurde festgestellt, dass es ein Problem bei Persistieren eines Objektes gab (Ressource Beschreibung), welches nicht auf die Datenbank gespeichert wurde. Dieses Fehlverhalten muss überprüft werden. d.h. Test ist nicht erfolgreich.

Table 6: Systemtest Datenspeicherung (Datenbank)

3.2 Broker

Die Funktionalität des Brokers wurde nicht getestet, da dieser während der Projektphase nicht realisiert werden konnte.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Projektplanung

19. Mai 2009

Abstract

Dieses Dokument beinhaltet die Projektplanung zu dieser Studienarbeit.

Inhalte dabei sind die Zielsetzungen, Projektorganisation, Terminplanung, Aufwandsschätzung, Projektplan, Risikomanagement, Qualitätssicherung und Projektdokumentation.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Projektplanung_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
04.03.2009	Dokument erstellt
19.05.2009	Dokument in Version 2 überarbeitet
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument gibt einen Aufschluss über die Projektplanung zur Studienarbeit.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Inhaltlich enthält dieses Dokument alle wesentlichen Informationen für die Verwaltung dieses Projektes.

2 Zielsetzungen

Für die Planung dieses Projektes sind folgende Ziele seitens des Projektverantwortlichen definiert worden:

- Fristgerechte Abgabe der Studienarbeit gemäss Terminvorgaben HSR
- Entwickeln eines funktionsfähigen Windows Druckertreibers um Bilddaten/PDFs zu erzeugen
- Entwickeln einer funktionsfähigen Client-Server Lösung für das Senden von Daten an ein DICOM System
- Konfiguration und Einsatz eines Build-Systems
- Einsatz von Systemtests (Unit-Tests, CI-Tests)
- Erstellen aller notwendigen Projektdokumente

3 Projektorganisation

Dieser Abschnitt beschreibt die Art und Weise wie die Zusammenarbeit in diesem Projekt geregelt wird. Es wird zwischen Aufbau- und Ablauforganisation unterschieden.

3.1 Aufbauorganisation

Das Projektteam setzt sich aus folgenden Personen zusammen:

Rolle	**Personen**
Verantwortlicher	Doering, Axel
Student	Vetter, Florian
Gegenleser	[Nicht definiert]
Experte	[Nicht definiert]
Industriepartner	[Nicht definiert]

Table 1: Aufbauorganisation

Die Rolle des Projektmanagers wird durch den Student wahrgenommen. Der Verantwortliche ist der Auftraggeber und kontrolliert die Durchführung der Studienarbeit.

3.2 Ablauforganisation

Für die Durchführung der Studienarbeit wird die Projektaufgabe in kleinere, überschaubare Teilaufgaben unterteilt. Diese können in einer bestimmten Reihenfolge, teilweise auch parallel ausgeführt werden.

Folgende Tätigkeiten zugeteilt an eine entsprechende Teilaufgabe sind während der Durchführung geplant:

Teilaufgabe	**Tätigkeiten**
Analyse	Aufgabenstellung analysieren Risikoanalyse
Entwurf / Design	Architekturdesign Prototyp Schnittstellen spezifizieren Protokolle definieren
Realisation	fertige Lösung programmieren (Ausbau des Prototyps)
Infrastruktur	Einrichten der persönlichen IDE Einrichten der Testumgebung Einrichten der Buildumgebung Einrichten des CI-Testing Dokumentation Infrastruktur
Dokumentation	Vorlagen erstellen Wiki (trac) Projektdokumente erstellen Dokumentation abschliessen
Testen	Schreiben von Testszenarios Durchführen von Systemtests: • Unit-Tests • CI-Tests Bug-tracking / Defect-tracking (um Übersicht zu behalten → alle Fehler der SW in System eintragen und risikobasierte Abschätzungen zu machen.)
Einführung	Lösung in Betrieb nehmen
Administration	Projektmanagement Projektplanung Sitzungen / Reviews

Table 2: Ablauforganisation

4 Terminplanung

Die nachfolgenden Termine sind für die Bearbeitung der Studienarbeit relevant.

Ergänzende Informationen können aus der HSR Webseite (HSR intern) entnommen werden, siehe Intranet → Bachelor → Informatik → Diplom-, Bachelor- und Studienarbeiten.

4.1 Durchführungszeit

Die Studienarbeit wird im Frühjahrssemester 2009 an der HSR durchgeführt. Dabei dauert das Semester vom 16.02.2009 – 13.09.2009.

4.2 Projektstart / Projektende

Beginn der Studienarbeit und Ausgabe der Aufgabenstellung durch die Betreuer ist Montag 16.02.2009. Ende der Studienarbeit ist der Fr 29.05.2009.

4.3 Wichtige Termine

Zwischen der Woche 8 und 9 ist an der HSR 1 Woche unterrichtsfrei, da die Osterfeiertage stattfinden. Diese Woche kann sehr gut genutzt werden um mit den Arbeiten weiter voran zu kommen. Es werden keine Sitzungen eingeplant.

Am Fr 29.05.2009 ist der Bericht bis spätestens 17.00Uhr dem Betreuer abzugeben. Zusätzlich muss eine Kurzbeschreibung der Studienarbeit als PDF an C. Furrer, cfurrer(at)hsr.ch mit entsprechendem Formular (Kurzfassung Studienarbeit) zugestellt werden.

4.4 Projektbesprechungen

Während dieser Studienarbeit sind wöchentliche Sitzungen eingeplant. Sie haben das Ziel den Arbeitsverlauf der Studienarbeit zu überwachen und geben dem Studenten die Möglichkeit zum Gedankenaustausch mit dem Dozenten. Die genauen Daten werden jeweils per Ende der Sitzung für die kommende Woche festgelegt.

Woche	Dauer	Sitzungsdatum	Sitzungsart
1	16.02.2009 – 22.02.2009	20.02.2009	Telefonbesprechung
2	23.02.2009 – 01.03.2009	27.02.2009	Telefonbesprechung
3	02.03.2009 – 08.03.2009	06.03.2009	Sitzung
4	09.03.2009 – 15.03.2009	13.03.2009	Sitzung
5	16.03.2009 – 22.03.2009	20.03.2009	Sitzung
6	23.03.2009 – 29.03.2009	27.03.2009	Telefonbesprechung
7	30.03.2009 – 05.04.2009	03.04.2009	Sitzung
8	06.04.2009 – 12.04.2009	10.04.2009	keine Sitzung, da Karfreitag
9	13.04.2009 – 19.04.2009	17.04.2009	Sitzung
10	20.04.2009 – 26.04.2009	24.04.2009	Sitzung
11	27.04.2009 – 03.05.2009	01.05.2009	Sitzung
12	04.05.2009 – 10.05.2009	08.05.2009	Sitzung
13	11.05.2009 – 17.05.2009	15.05.2009	Sitzung
14	18.05.2009 – 24.05.2009	22.05.2009	Sitzung
15	25.05.2009 – 29.05.2009	27.05.2009	Sitzung

Table 3: Projektbesprechungen Terminübersicht

5 Eingesetztes Vorgehensmodell zur Softwareentwicklung

Für die Studienarbeit wird ein agiles Projektvorgehen (Prozessmodell) gewählt. Dies hat zu Folge, dass die Softwareentwicklung agil abläuft, d.h. die zu entwickelnde Softwarelösung und die Zufriedenheit des Benutzers stehen im Mittelpunkt. Es wird ein Prototyp erstellt, welcher laufend um Funktionalität erweitert wird.

6 Aufwandsschätzung

Die Studienarbeit wird mit total 8 ECTS Punkte honoriert, d.h. es sind 8h Vorlesungszeit und im Minimum zusätzlich 8h im Selbststudium dafür aufzuwenden. Diese ergibt einen durchschnittlichen Arbeitsaufwand von etwa 16h pro Woche. Bei einer Projektdauer von 15 Wochen kann man mit ungefähr 240h Arbeitsstunden rechnen, was in etwa 30 Manntage (MT) gleichzusetzen ist (1 Manntag = 8h).

6.1 Nach Projektphasen

Da der effektive Arbeitsaufwand nicht beziffert werden kann (Menge unbekannt), ist eine Aufteilung der Arbeit auf entsprechende Projektphasen nur grob abgeschätzt worden. Diese muss während dem Projektverlauf entsprechend angepasst werden.

- Einarbeitungsphase (ca. 5MT)
- Analysephase (ca. 3MT)
- Entwurfsphase (ca. 7MT)
- Realisierungsphase (ca. 5MT)
- Übergabephase (ca. 10MT)

6.2 Nach Arbeitspakete

Für die Gliederung der Tätigkeiten wurden folgende Arbeitspakete vorgesehen und sind mit entsprechender Aufwandsschätzung verbunden:

Name	**Aufwand**	**Tätigkeiten**
Druckertreiber	10MT	Entwickeln eines virtuellen Druckertreibers Erstellen von PDFs Erstelle von Bilder (Bilddateien) Systemtests Builds vom Druckertreiber erstellen
Client/Server System	14MT	Proxy für Schnittstelle Druckertreiber/DICOM Archiv Übertragungsprotokoll (Client Server) Lokaler Cache für Datenzwischenspeicherung Konfigurationsverwaltung Systemtests (Bsp. Unit-Tests) Builds erstellen der Softwarelösung
Infrastruktur	6MT	Testumgebung aufbauen Build-System einrichten CI-Tests (laufend)

Table 4: Arbeitspakete

Gegeben durch die Situation einer Einzelarbeit ist die Manpower, mit nur einer einzigen Person, für die Entwicklung der Softwarelösung sehr beschränkt. Dies hat zur Folge, dass nicht parallel an den einzelnen Komponenten gearbeitet werden kann, sondern die Programmierung nach Prioritäten angegangen werden muss.

7 Projektplan

Der Projektplan liefert eine Übersicht aller Tätigkeiten zu den einzelnen Phasen des Projektes. Wichtige Ereignisse wurden als Milestones definiert.

7.1 Projektphasen

Ausgehend von einem zu lösenden Problem, kann eine allgemeine Unterteilung in folgende Phasen gemacht werden:

- Analysephase
  - Bestandsaufnahme der aktuellen Situation
- Konzeptionsphase
  - Ziele definiert und Alternativen bewertet
- Realisierungsphase
  - Entwicklung eines Prototypen
  - Ausgestaltung des zukünftigen Systems anhand des Prototypen.
- Übergabephase
  - System wird Endbenutzer übergeben, erstellen der Projektdokumentation

7.2 Milestones

Folgende Milestones wurden für das Projekt vorgesehen:

MS	Datum	Projekt-woche	Beschreibung
1	01.03.2009	Wo2	Einrichten persönliche Arbeitsumgebung (Arbeitsplatz, Repository, Wiki)
2	08.03.2009	Wo3	Abschluss Einrichtung Infrastruktur (Server, Testserver, Build system)
3	15.03.2009	Wo4	Abschluss Analysephase
4	05.04.2009	Wo7	Ende Entwurfsphase
5	26.04.2009	Wo10	Prototyp fertig
6	11.05.2009	Wo13	Ende Realisierungsphase
7	22.05.2009	Wo14	Dokumentation im Status „Review“ abgeschlossen
8	27.05.2009	Wo15	Dokumentation im Status „Final“ abgeschlossen
9	29.05.2009	Wo15	Abgabe Beschreibung der Studienarbeit
10	29.05.2009	Wo15	Abgabe Studienarbeit (Dokumentation und Software auf CD-ROM) um 17:00 Uhr

Table 5: Milestones

7.3 Projektplan Ganntt-Chart

Für eine grafische Übersicht der Projektplanung in einem Ganntt-Chart wird auf den *Projektplan* verwiesen, welcher im Anhang der Dokumentation zu finden ist

7.4 Kritikpunkte

Es wird festgehalten, dass für die DICOM spezifischen Funktionalitäten sehr wenig Dokumentation bezüglich dem einzusetzenden DICOM-Toolkit von dcm4che.org vorhanden ist, bzw. diese Dokumentation sehr dürftig ausfällt. Dies bedeutet, dass bei der Realisierung/Programmierung der Komponenten entsprechendes Know-How erarbeitet werden muss.

8 Risikomanagement

Im Risikomanagement macht man sich Gedanken über mögliche Risiken, welche in der durchzuführenden Projektarbeit auftreten können. Diese werden aufgelistet, bewertet und ein möglicher resultierender Schaden abgeschätzt. Für jedes Risiko werden anschliessend entsprechende Massnahmen festgelegt (siehe 8.7 Massnahmenplan).

8.1 Identifikation der grössten Risiken

Für dieses Projekt gibt es 2 grosse Risiken:

- Treiberprogrammierung
- Einarbeitung in den DICOM-Standard

Da die Einarbeitung in den DICOM-Standard zwingend notwendig ist, muss für diese Tätigkeit entsprechende Zeit eingerechnet werden.

Bezüglich dem Risiko Treiberprogrammierung kann gesagt werden, dass wenn dieser Arbeitsschritt nicht ausgeführt werden kann, z. B. wegen mangelndem Know-How, muss ein anderer Weg die Softwarelösung zu realisieren gesucht werden.

8.2 Risikoanalyse Projekt

Nr	Risiko	Schaden	Eintrittswahr-scheinlichkeit	Massnahmen
1	DICOM ist unbekanntes Thema	30h	50.00%	Genug Zeit ein berechnen für Einarbeitung ins Thema DICOM
2	Hohe Komplexität der Aufgabenstellung	24h	40.00%	Viel Fragen stellen, Lesen von Fachartikel, evtl. Hilfe in Anspruch nehmen, umfangreiche Dokumentation erstellen für besseres Verständnis
3	Knappes Zeitbudget	4h	25.00%	Priorisiereung der Arbeitspakete, Unterteilung der Funktionalität in „must have“ und „nice to have“. Funktionalität einschränken durch integrieren anderer, schon bestehender Systeme (z.B. bestehende Druckertreiber).
4	Eingesetzte Technologie erweist sich als Stolperstein	15h	20.00%	Wenn möglich auf bestehende, bekannte Technologie setzen
5	Qualität lässt nach, da Umfang zu gross ist	20h	20.00%	Umfang der Funktionalität zurückschrauben. Einsetzen von Systemtests

Table 6: Risikoanalyse Projekt

8.3 Risikoanalyse Softwarelösung

Nr	Risiko	Schaden	Eintrittswahr-scheinlichkeit	Massnahmen
6	Treiberprogrammierung ist zu aufwändig	10h	50.00%	Umschauen, ob ev. Nicht ein bestehender Treiber vorhanden ist, welche Funktionalität enthält
7	Lösung läuft nicht auf Endgerät	4h	50.00%	Mindestanforderungen an HW/SW definieren, damit Kompatibilität gewährleistet ist.
8	Treffen einer falschen Architektur-Entscheidungen	20h	20.00%	Besprechung/Diskussion des gewählten Architekturdesigns mit Dozent, Designpatterns einsetzen
9	Lösung ist für den Endbenutzer schwer bedienbar	10h	20.00%	Lösung so entwickeln/entwerfen, dass diese intuitiv zu bedienen ist, d.h. einfach, leicht verständlich). Zusätzlich Benutzerdokumentation erstellen .

Table 7: Risikoanalyse Entwickelnde Softwarelösung

8.4 Risikoanalyse Infrastruktur

Nr	Risiko	Schaden	Eintrittswahr-scheinlichkeit	Massnahmen
10	Datenverlust	40h	10.00%	Regelmässiges Backup, Zentraler Ablageort für Daten in ein Repository, Antivirenlösung einsetzen
11	Defekte HW	8h	5.00%	Alternativer Arbeitsplatz, Ersatzgeräte bereithalten
12	Sicherheitslücken machen System unbrauchbar	16h	1.00%	Neuste Versionen der Tools verwenden, inkl. Aktuelle Patches installieren

Table 8: Risikoanalyse Infrastruktur

8.5 Risikoanalyse Personal

Nr	Risiko	Schaden	Eintrittswahr-scheinlichkeit	Massnahmen
13	Krankheit	24h	50.00%	Zeitreserven einplanen
14	Unfall	80h	10.00%	Zeitreserven einplanen

Table 9: Risikoanalyse Personal

8.6 Risikoauswertung

Folgender Schaden in Stunden kann gemäss Risikoanalyse erwartet werden:

Risiko	Total Schaden pro Kategorie	Bemerkungen
Projekt	33h	geschätzt
Softwarelösung	13h	geschätzt
Infrastruktur	5h	geschätzt
Personal	20h	geschätzt
Total	71h	Entspricht ca. 9MT, Zeitreserven ein berechnen um allfällige Risiken durch Zeitverzögerung zu kompensieren.

Table 10: Risikoauswertung

8.6.1 Erläuterung

Die Risikoauswertung zeigt auf, dass ca. 9MT Zeitreserve eingerechnet werden muss. 9 MT entsprechen dabei in etwa 1/3 der gesamten Projektdauer. Es ist nicht möglich so viel Zeit einzuplanen, darum würden sich die abgeschätzten Risiken negativ auf den Zeitplan auswirken, bzw. es treten Verzögerungen in der Projektdurchführung auf. Dies wiederum führt zu Nichteinhalten, bzw. Verschiebung der Termine/Milestones im Bezug auf die Abgabe dieser Arbeit.

8.7 Massnahmenplan

Die bei der Risikoanalyse aufgelisteten Massnahmen stellen mögliche Alternativen dar, bzw. zeigen ob das Szenario funktioniert.

Auch sollte man sich im Voraus überlegen, ob mögliche Abkürzungen im Projekt genommen werden könnten.

Am Beispiel der Treiberentwicklung stellt sich heraus, dass mit einem viel höheren Aufwand zu rechnen ist. Hier könnte man einen bestehenden Druckertreiber verwendet, welche die gewünschte Funktionalität bereits implementiert hat um PDF / Bilder zu erzeugen.

Generell sollte die zu entwickelnde Lösung möglichst unabhängig untereinander sein, d.h. Schnittstellen mit entsprechenden Funktionalitäten/Verantwortlichkeiten definieren.

Als Beispiel: Dienst schreiben, welcher nun auf eine bestehenden Stelle im Dateisystem schaut ob Daten vorhanden sind. Falls gefunden, dann wird dies an den DICOM Server geschickt.

9 Qualitätssicherung

Qualität ist eines der zentralen Punkte einer jeden Lösung. Um die Qualität der fertigen Arbeit auf einem konstanten, hohen Niveau zu halten, werden während der Studienarbeit verschiedenste Tätigkeiten, oder Hilfsmittel eingesetzte:

- Projektbesprechungen (wöchentlich um Stand der Arbeiten festzuhalten)
- Projektkontrollsitzungen (Projektreviews, meistens kombiniert mit Projektbesprechung)
- Planungskontrolle (durch Projektleiter durchzuführen, in Bezug auf die Termine, sowie entsprechendes Einleiten notwendiger Massnahmen bei auftretenden Problemen.)
- Unit-Tests (X-Unit) bei Softwareentwicklung
- Testgetrieben programmieren (TDD = Test driven development)
- Continuous Integration Tests (siehe Build-System)
- Betrieb eines Build-Systems, welche die Software beim Einchecken in Mainline des Repositorys Projekt neu buildet. Dabei laufen automatisch sogenannte Integrations- und Systemtests ab, welche dem Entwickler Feedback über den Build-Status geben.
- Einsatz eines Versionsverwaltungssystems (Subversion)
- Programmierung nach den an der HSR vermittelten Kenntnisse

9.1 Anmerkungen zur Qualitätssicherung

Die Punkte testgetriebene Programmierung, Unit-Testing und Continuous Integration kamen während der Entwicklungsphase dieser Studienarbeit zu kurz, oder konnten gar nicht richtig eingesetzt werden wie ursprünglich vorgesehen. Grund dafür war einerseits die fehlende Zeit dazu, bzw. die Komplexität der DICOM Funktionalität.

10 Projektdokumentation

Während dieser Studienarbeit werden verschiedenste Dokumente erstellt. Sprache dieser Dokumente ist auf Deutsch festgelegt.

Die Dokumentation muss allgemein als auch formellen Ansprüchen genügen.

10.1 Allgemeine Anforderungen

Folgende Anforderungen werden an die Dokumentation gestellt:

- Vollständigkeit, d.h. es dürfen keine wichtigen Punkte fehlen.
- Übersichtlichkeit, d.h. die Beschreibungen müssen klar untergliedert werden und schnell auffindbar sein.
- Verständlichkeit, d.h. Dritte müssen sich anhand der Dokumentation schnell und selbstständig in ein Thema einarbeiten können.
- Dokumente sollen formale Anforderungen zumindest Gliederung, Titel, Autor, Erstellungsdatum und Dokumentestatus (DRAFT, REVIEW, FINAL) aufweisen.

- Für jedes Dokument soll eine PDF-Version vorliegen, damit diese auf möglichst allen Plattformen angeschaut werden kann.

10.2 Zu erstellende Dokumente

Dokumentname	Leser	Beschreibung / Inhalt
Anforderungsanalyse	Entwickler, Auftraggeber	Ist/Soll, UC, Szenarios,
Anforderungsspezifikation	Entwickler, Auftraggeber	(=Software Request Specification / Pflichtenheft -> Vertrag zwischen Entwickler und Auftraggeber) Funktionale- Nichtfunktionale Anforderungen, mit Nr für Verfolgbarkeit der Anforderungen
Architekturdesign	Entwickler	Beschreibung und Überlegungen (Klassendesign, Entwurfsmuster) Entwurfsdokumentation, Architektur, Designgedanken
Systemtests / Software Verifizierungsplan	Entwickler	Plan wie Software zu testen ist (was, wie, Ergebnis). Nachweis der Durchführung, Erkenntnis daraus
Configurations Management Plan	Entwickler	SVN Repository, Buildumgebung – Vorgang, Installation, Konfigurationen
Projektplanung / Projektmonitoring	IT-Management	Angaben zum Projektverlauf, Projektplanung, Projektplan
Persönlicher Bericht	Alle	enthält Nachweis, dass Student Plan/Arbeit durchgeführt hat
Betreiberdokumentation	Entwickler, IT-Abteilung	muss für den Systembetrieb im Rechenzentrum vorgelegt werden, enthält Installationsanleitung
Benutzerdokumentation	Benutzer, Fachabteilung	Beschreibung des Leistungsumfanges des Systems, inkl. Sonderfälle und Schnittstellen. (Infos, was Benutzer alles wissen muss) Schulung und als Nachschlagewerk (Infos zur Bedienung der Software)
Einführung in DICOM	Entwickler	Beschreibt die wesentlichen Eigenschaften von DICOM
Glossar	Entiwckler, Benutzer	Abkürzungsverzeichnis, Begriffserklärung

Table 11: Übersicht der zu erstellenden Dokumente

10.3 Dokumente nach Projektphasen

Hier werden die Dokumente den einzelnen Phasen zugeordnet, inkl. Stand des Dokumentes.

Projektphase	**Dokument**	**Stand**
Analysephase	Anforderungsanalyse	DRAFT
	Anforderungsspezifikation	DRAFT
Konzeptionsphase	Architekturdesign	DRAFT
Realisierungsphase	Architekturdesign	DRAFT
	Betreiberdokumentation / Installationsanleitung	DRAFT
	Software Verifizierungsplan	DRAFT
Testphase	Systemtests	DRAFT
Übergabephase	Bericht	FINAL
	Anwenderdokumentation / Benutzerhandbuch	REVIEW
	alle Dokumente	REVIEW
	alle Dokumente	FINAL

Table 12: Dokumente nach Projektphasen

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Glossar

19. Mai 2009

Abstract

Dieses Dokument beinhaltet den Glossar zu dieser Studienarbeit.

Inhalt dabei sind alle verwendeten Fachwörter oder Abkürzungen mit jeweiliger Erklärung.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Glossar_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	**Beschreibung**
05.03.2009	Dokument erstellt
19.05.2009	Dokument in Version 2 überarbeitet
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument beinhaltet den Glossar zu dieser Studienarbeit.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Dieses Dokument beschreibt alle Begriffe und Abkürzungen, welche in der Studienarbeit vorkommen.

2 Begriffe

Begriff	Beschreibung
AE	Ist die Abkürzung von Application Entity und bezeichnet einen lokalen oder entfernten DICOM Service.
Broker	Ist ein Vermittler zwischen 2 Parteien, d.h. er kümmert sich um bei ihm abgegebene Aufträge (Entgegenmahme und Weiterleitung an die entsprechende Stellen) und führ Buch über die Zustände der Aufträge.
Client	Programm, welches meistens auf der Arbeitsstation des Benutzers installiert ist und einen Service von einem Server in Anspruch nimmt.
dcm4che	Ist eine Sammlung von Open Source Anwendungen und Hilfsmittel (utilities) für die Gesundheitsindustrie, welche in Java entwickelt wurde. Weitere Informationen siehe: http://www.dcm4che.org/
dcm4che2 Toolkit	Toolkit von dcm4che, das eine nützliche Sammlung von kleinen Programmen anbietet, die spezifische DICOM-Funktionalitäten zur Verfügung stellen.
DICOM	Digital Imaging and Communication in Medicine. DICOM ist ein offener Standard zum Austausch von Informationen in der Medizin.
DICOM-Archiv	Ist eine elektronische Datenbank über Patienten.
DICOM Konformität	Dieser Begriff sagt aus, dass die DICOM-Daten gemäss dem DICOM-Standard richtig erstellt wurden.
DICOM-Server	Ist ein Computer, welche Services gemäss DICOM Standard Clients zu Verfügung stellt.
DICOM viewer	Sind spezielle Bildbetrachter, mit welchen DICOM Dateien angeschaut werden können.
HL7	Health Level 7 ist eine Gruppe internationaler Standards, welche den Austausch von Daten zwischen Organisationen im Gesundheitswesen und deren Computersystemen regelt.
HSQLDB	Ist eine in Java geschreiebene relationale SQL Datenbank. Weitere Informationen siehe: http://hsqldb.org
JPA	Java Persistent API Ist eine Schnittstelle für Java-Anwendungen, welche die Übertragung von Objekten zu Datenbankeinträgen vereinfacht. Weitere Informationen siehe: http://java.sun.com/javaee/technologies/persistence.jsp
Modalität	Ist ein in der Medizin bilderfassendes Gerät.
PACS	Picture Archiving and Communication System ist ein Bildarchivierungs- sowie Kommunikationssystem, welches in der Medizin eingesetzt wird. Es setzt auf digitale Rechner und Computernetzwerke auf.
Storage Commitment	Mittels dem DICOM Storage Commitment Service werden Bilddaten auf der lokalen Festplatte erst nach Bestätigung der Archivierung durch das Archiv gelöscht.

SCP	Service Class Provider DICOM Begriff welcher sagt, dass diese Instanz einen Service anbietet. (kann mit der Rolle eines Servers gleichgesetzt werden)
SCU	Service Class User DICOM Begriff welcher sagt, dass diese Instanz einen Service benutzt. (kann mit der Rolle eines Clients gleichgesetzt werden)
Study Instance UID	DICOM Begriff für einen eindeutigen Bezeichner einer Studie

Table 1: Begriffe

3 Abkürzungen

Abkürzungen	Bedeutung
AE	Application Entity
AET	Application Entity Title
CT	Computer Tomograph
CUID	Class Unique Identifier
DICOM	Digital Imaging and Communication in Medicine
EJB	Enterprise Java Beans
HL7	Health Level 7
HSQLDB	HyperSQL DataBase
IHE	Integrating the Healthcare Enterprise
JDK	Java Developer Kit
JMS	Java Messaging Queue
JPA	Java Persistent API
JPEG	Joint Photographic Experts Group
PACS	Picture Archiving and Communication System
SC	Service Class
SCP	Service Class Provider
SCU	Service Class User
UID	Unique Identifier

Table 2: Abkürzungen

Tabellenverzeichnis

Abbildungsverzeichnis

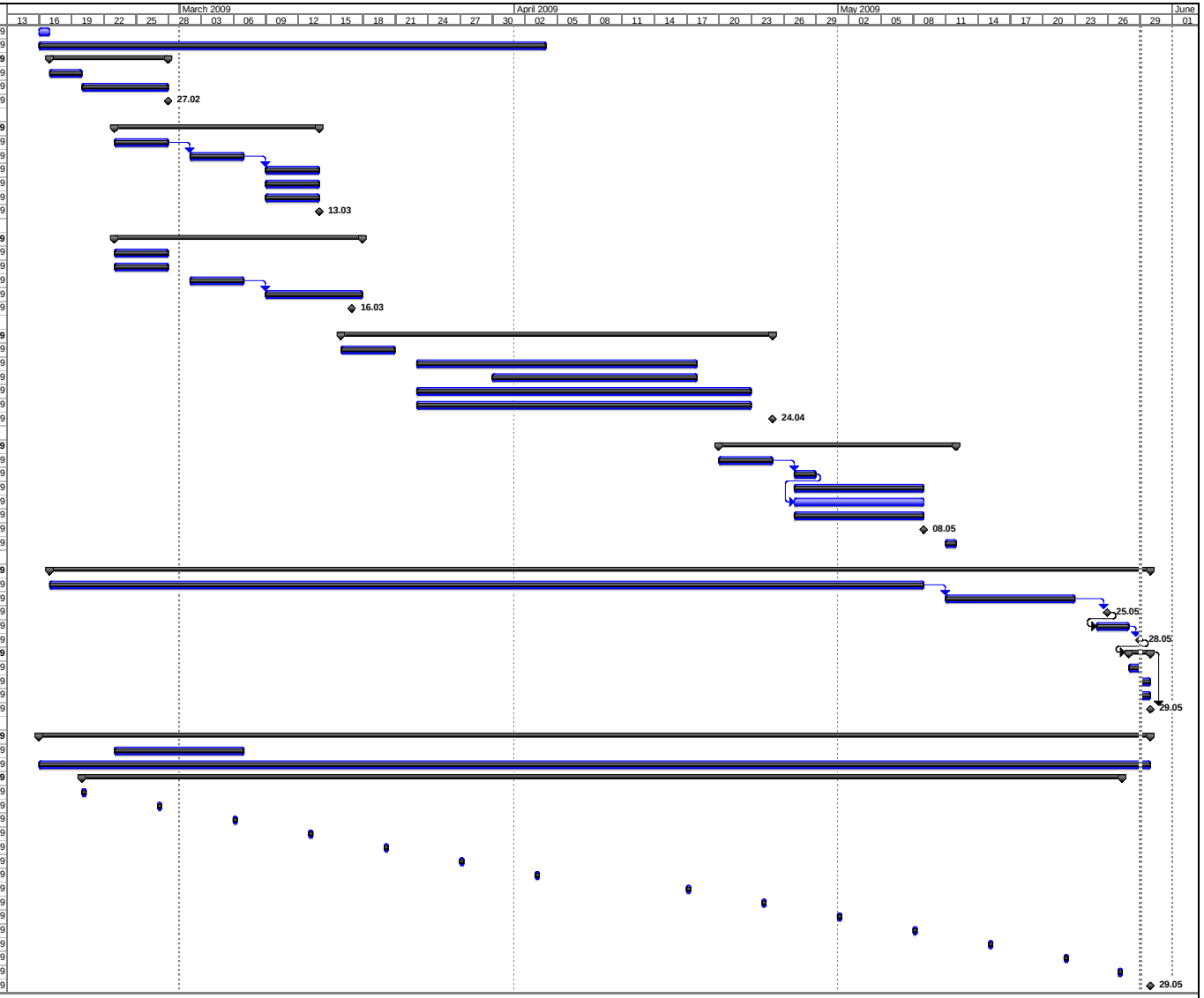
Literaturverzeichnis

HSR Hochschule für Technik Rapperswil
Studienarbeit, FS 2009

Anhang

- Projektplan
- Projektmonitoring
- Protokolle

ID	Task Name	Duration	Start	Finish
1	Beginn Studienarbeit	1 day?	Mon 16.02.09	Mon 16.02.09
2	DICOM (Einarbeit Einlesen in diesen Standard)	35 days?	Mon 16.02.09	Fri 03.04.09
3	**Persönliche Arbeitsumgebung**	**9 days?**	**Tue 17.02.09**	**Fri 27.02.09**
4	Arbeitsplatz einrichten	3 days?	Tue 17.02.09	Thu 19.02.09
5	Trac (SVN-Repository und Wiki)	6 days?	Fri 20.02.09	Fri 27.02.09
6	Arbeitsumgebung eingerichtet	1 day?	Fri 27.02.09	Fri 27.02.09
7				
8	**Infrastruktur**	**15 days?**	**Mon 23.02.09**	**Fri 13.03.09**
9	virtuelle Arbeitsumgebung einrichten (Sun xVM VirtualBox)	5 days	Mon 23.02.09	Fri 27.02.09
10	DICOM-Server	5 days	Mon 02.03.09	Fri 06.03.09
11	DICOM-Toolkit	5 days	Mon 09.03.09	Fri 13.03.09
12	Build-Server	5 days	Mon 09.03.09	Fri 13.03.09
13	Application-Server	5 days	Mon 09.03.09	Fri 13.03.09
14	MS: Infrastruktur eingerichtet	1 day?	Fri 13.03.09	Fri 13.03.09
15				
16	**Analyse**	**17 days?**	**Mon 23.02.09**	**Tue 17.03.09**
17	Anforderungsanalyse	5 days?	Mon 23.02.09	Fri 27.02.09
18	Risikoanalyse	5 days?	Mon 23.02.09	Fri 27.02.09
19	User Szenarios	5 days?	Mon 02.03.09	Fri 06.03.09
20	Use Cases	7 days?	Mon 09.03.09	Tue 17.03.09
21	MS: Analyse Abgeschlossen	1 day?	Mon 16.03.09	Mon 16.03.09
22				
23	**Prototyp**	**30 days?**	**Mon 16.03.09**	**Fri 24.04.09**
24	Funktionaler Prototyp (Testet Funktionalität der eingesetzten Komponenten)	5 days?	Mon 16.03.09	Fri 20.03.09
25	Anforderungsspezifikation	20 days?	Mon 23.03.09	Fri 17.04.09
26	Architektur-Design	15 days?	Mon 30.03.09	Fri 17.04.09
27	Programmierung Client	23 days	Mon 23.03.09	Wed 22.04.09
28	Testen	23 days?	Mon 23.03.09	Wed 22.04.09
29	MS: Prototyp fertig	20 days?	Mon 30.03.09	Fri 24.04.09
30				
31	**Softwarelösung**	**16 days?**	**Mon 20.04.09**	**Mon 11.05.09**
32	Softwareprojekt einrichten	5 days?	Mon 20.04.09	Fri 24.04.09
33	Überführung Prototyp in Softwarelösung	2 days?	Mon 27.04.09	Tue 28.04.09
34	Programmierung Ri4idd2dicom Client	10 days?	Mon 27.04.09	Fri 08.05.09
35	Programmierung Ri4idd2dicom Broker	10 days?	Mon 27.04.09	Fri 08.05.09
36	Testen	10 days?	Mon 27.04.09	Fri 08.05.09
37	MS: Softwareprojektstand einfrieren	1 day?	Fri 08.05.09	Fri 08.05.09
38	Erstellen Build aus Softwareprojektstand	1 day?	Mon 11.05.09	Mon 11.05.09
39				
40	**Dokumentation**	**74 days?**	**Tue 17.02.09**	**Fri 29.05.09**
41	Arbeit im Status Entwurf	59 days?	Tue 17.02.09	Fri 08.05.09
42	Arbeit im Status Review	10 days?	Mon 11.05.09	Fri 22.05.09
43	MS: Dokumentatoin Status Review	1 day?	Mon 25.05.09	Mon 25.05.09
44	Arbeit im Status Final	3 days?	Mon 25.05.09	Wed 27.05.09
45	MS: Dokumentation Status Final	1 day?	Thu 28.05.09	Thu 28.05.09
46	**abschliessende Arbeiten**	**2 days?**	**Thu 28.05.09**	**Fri 29.05.09**
47	Kurzfassung der Sudienarbeit per E-Mail zusenden	1 day?	Thu 28.05.09	Thu 28.05.09
48	CD-ROM brennen / beschriften	1 day?	Fri 29.05.09	Fri 29.05.09
49	Dokumentation ausdrucken und heften	1 day?	Fri 29.05.09	Fri 29.05.09
50	MS: Abgabe Dokumentation	1 day?	Fri 29.05.09	Fri 29.05.09
51				
52	**Projektmanagement**	**75 days?**	**Mon 16.02.09**	**Fri 29.05.09**
53	Projektplanung	10 days?	Mon 23.02.09	Fri 06.03.09
54	Administration (Zeitmanagement, Projektleitung, Projektauswertung)	75 days?	Mon 16.02.09	Fri 29.05.09
55	**Meetings**	**68.13 days**	**Fri 20.02.09**	**Wed 27.05.09**
56	Meeting Wo1 (Kick-Off Meeting)	1 hr	Fri 20.02.09	Fri 20.02.09
57	Meeting Wo2	1 hr	Fri 27.02.09	Fri 27.02.09
58	Meeting Wo3	1 hr	Fri 06.03.09	Fri 06.03.09
59	Meeting Wo4	1 hr	Fri 13.03.09	Fri 13.03.09
60	Meeting Wo5	1 hr	Fri 20.03.09	Fri 20.03.09
61	Meeting Wo6	1 hr	Fri 27.03.09	Fri 27.03.09
62	Meeting Wo7	1 hr	Fri 03.04.09	Fri 03.04.09
63	Meeting Wo9	1 hr	Fri 17.04.09	Fri 17.04.09
64	Meeting Wo10	1 hr	Fri 24.04.09	Fri 24.04.09
65	Meeting Wo11	1 hr	Fri 01.05.09	Fri 01.05.09
66	Meeting Wo12	1 hr	Fri 08.05.09	Fri 08.05.09
67	Meeting Wo13	1 hr	Fri 15.05.09	Fri 15.05.09
68	Meeting Wo14	1 hr	Fri 22.05.09	Fri 22.05.09
69	Meeting Wo15 (Schlussmeeting)	1 hr	Wed 27.05.09	Wed 27.05.09
70	MS: Ende Studienarbeit	1 day	Fri 29.05.09	Fri 29.05.09



Project: Studienarbeit ri4idd2dicom
Date: Fri 29.05.09

Task | Progress | Summary | External Tasks | Deadline
Split | Milestone | Project Summary | External Milestone

Projektmonitoring

19. Mai 2009

Abstract

Dieses Dokument beinhaltet das Projektmonitoring zu dieser Studienarbeit.

Inhalte dabei sind das Zeitmanagement mit den Zeitdiagrammen und eine Übersicht aller durchgeführten Sitzungen, sowie eine Review zum Projektverlauf.

Autor	Florian Vetter
Organisation	HSR Hochschule für Technik Rapperswil
Projektname	ri4idd2dicom
Projekt / Thema	Studienarbeit (SA) Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die Medizinische IT-Infrastruktur retrofittable interface for integrating diagnosis devices into medical it infrastructure
Dokumentname	Projektmonitoring_v2.0.odt
Version	2.0
Status	FINAL

Bearbeitungsverlauf

Datum	Beschreibung
12.05.2009	Dokument erstellt
19.05.2009	Dokument in Version 2 überarbeitet
22.05.2009	Dokument bereit für Review
29.05.2009	Dokument Final

Inhaltsverzeichnis

1 Dokument

1.1 Zweck

Dieses Dokument beinhaltet das Projektmonitoring zu dieser Studienarbeit.

1.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für die gesamte Studienarbeitsdauer.

1.3 Übersicht

Inhalte dieses Dokumentes sind das Zeitmanagement mit den Zeitdiagrammen und eine Übersicht aller durchgeführten Sitzungen.

2 Zeitmanagement

Während der Durchführung dieser Studienarbeit führte jedes Teammitglied ein persönliches Zeiterfassungsdokument. In diesem Dokument wurden alle erledigten Arbeiten mit Zeitaufwand und Zuordnung zu entsprechenden Kategorien festgehalten.

Die Studienarbeit geht von einem wöchentlichen Zeitaufwand von ca. 16 Arbeitsstunden aus. Dieser kann ja nach Aufwand von Woche zu Woche unterschiedlich hoch ausfallen.

2.1 Definition der Kategorien für Zeiterfassung

Damit eine quantitative Aussage über den Zeitaufwand der ausgeführten Arbeiten erstellt werden kann, wurden nachfolgende Kategorien definiert.

Kategorie	**Beschreibung**
Projektmanagement	Zeiterfassung, Planen, Administrative Arbeiten, Erstellen von Traktandenliste und Protokoll zu den Sitzungen
Meetings	Sitzungen, Telefonkonferenz
Dokumentation	Erstellen aller Dokumente
Analyse	Analysieren der Problemstellung
Recherche	Selbststudium in Thematik, notwendiges Know-How erarbeiten
Prototyp	Programmierung
Entwicklung	Programmierung
Infrastruktur	Hardware, Software, Konfiguration

Table 1: Definition Kategorien für Zeiterfassung

2.2 Zeitdiagramme

Im folgenden Abschnitt sind die einzelnen Diagramme jeder Teammitglieder aufgeführt. Sie geben einen Überblick über die aufgewendeten Stunden pro Woche im Vergleich zur Projektvorgabe und die Gesamtverteilung aller Arbeitsstunden auf die einzelnen Kategorien.

2.2.1 Zeitaufwand nach Kategorien

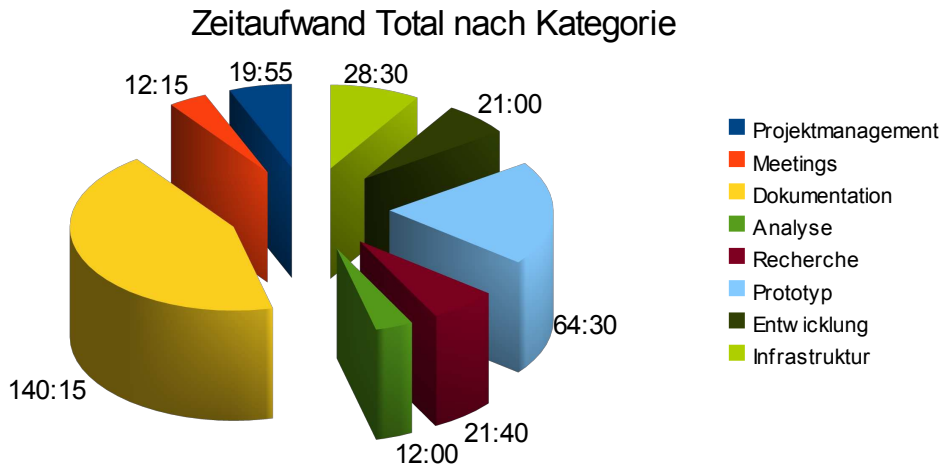


Illustration 1: Diagramm Zeitaufwand nach Kategorien

2.2.1.1 Erläuterung

Die meiste Zeit wurde für das Erstellen der Dokumentation aufgewendet (ca. 140h), gefolgt von der Entwicklung des Prototypen (ca. 65h) mit anschliessender Weiterentwicklung der Softwarelösung in einem dafür überführten Projekt (ca. 21h). An dritter Stelle folgen die Arbeiten für das Einrichten und Betreiben der Infrastruktur (beinhaltet Entwicklung- und Arbeitsumgebung mit den entsprechend dazugehörenden Servern) (ca. 29h).

Der Teil Infrastruktur umfasst alle Tätigkeiten zum Einrichten der Umgebung. Durch die Einarbeitung in ein neues Thema musste entsprechend Zeit aufgewendet werden, welche die Kategorie Recherche zusammenfasst (ca. 22h).

2.2.2 Zeitaufwand nach Wochen von Florian Vetter

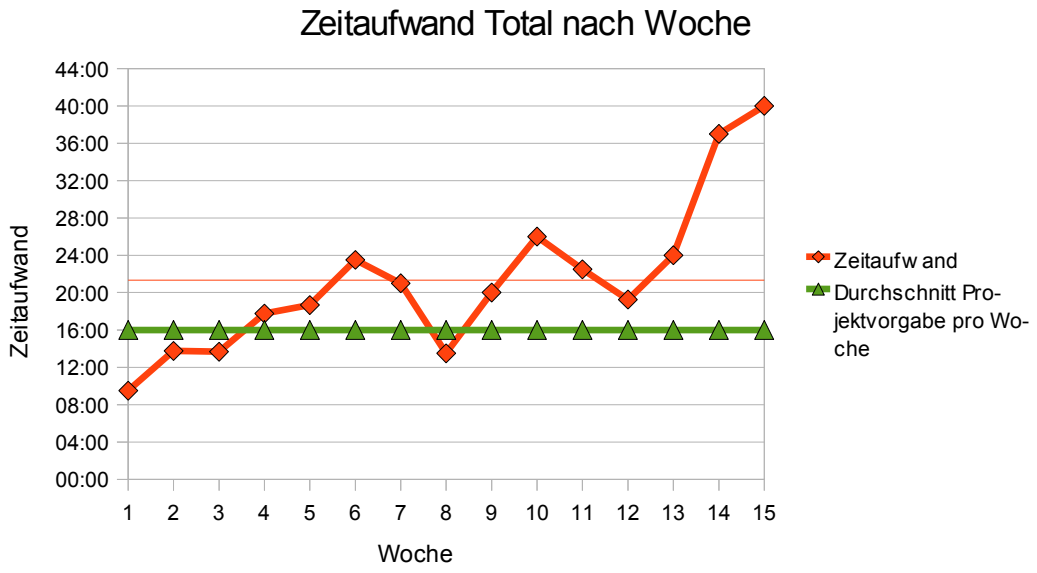


Illustration 2: Diagramm Zeitaufwand nach Wochen Florian Vetter

2.2.2.1 Erläuterung

Durch die agile Projektdurchführung wurde entsprechend Arbeitsschwerpunkt Zeiten aufgewendet. Man kann eine Steigung der Arbeitsstunden gegen Ende der Abgabe feststellen, da noch diverse Abschlussarbeiten durchzuführen waren (in den letzten 3 Wochen hauptsächlich Dokumentationsarbeiten. Der Stundeneinbruch in Woche 8 war auf die unterrichtsfreie Woche zurückzuführen, in welchen die Osterfeiertage waren.

Über die ganze Dauer der Projektdauer wurde durchschnittlich über 20 Arbeitsstunden pro Woche für diese Studienarbeit aufgewandt, was die durchschnittliche Projektvorgabe zeitlich um mehr als 4h pro Woche überschreitet.

2.3 Bemerkung Zeiterfassung

Eine detaillierte Erfassung der Arbeitszeiten (aufgeteilt nach Disziplinen entsprechend Projektplan) fehlt, da dies nicht vorgesehen war.

Aus dem Diagramm Zeitaufwand nach Kategorien kann heraus gelesen werden, dass für die einzelnen Disziplinen im Projektplan eine Kategorie zugeordnet werden kann. Ausgenommen davon sind die Kategorien Dokumentation, bzw. Projektmanagement und Meetings welche über die ganze Projektdauer durchgeführt wurden.

2.4 Sitzungen

Folgende Sitzungen wurden während dieser Studienarbeit durchgeführt:

Woche	Datum	Zeit	Teilnehmer	Ort / Bemerkungen
1	20.02.2009	08:00-09:00 Uhr	Team & Dozent	Telefonmeeting
2	27.02.2009	08:00-08:30 Uhr	Team & Dozent	Telefonmeeting
3	06.03.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
4	13.03.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
5	20.03.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
6	27.03.2009	08:00-08:30 Uhr	Team & Dozent	Telefonmeeting
7	03.04.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
9	17.04.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
10	24.04.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
11	01.05.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
12	08.05.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
13	15.05.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
14	22.05.2009	08:00-09:00 Uhr	Team & Dozent	HSR, IFS
15	27.05.2009	17:45-18:15 Uhr	Team & Dozent	HSR, IFS

Table 2: Sitzungen

Für jede Sitzung wurde im Voraus eine Traktandenliste, sowie ein Protokoll mit den Beschlüssen nach der Sitzung erstellt. Für Einsicht in die Protokolle wird auf den *Anhang* in der Dokumentation verwiesen.

3 Review Projektverlauf

Gegeben durch das Studienarbeitsthema musste für den Verlauf der Projektdauer eine Projektplanung erstellt werden. Durch das Arbeiten in unbekannten Arbeitsgebiete (Treiberprogrammierung, DICOM) konnte der effektive Arbeitsaufwand der zu entwickelnden Softwarelösung nur grob abgeschätzt werden. Die erstellte Projektplanung zu Beginn der Studienarbeit ging davon aus, dass alle Tätigkeiten während den 15 Wochen Projektdauer durchzuführen sein sollten.

Im Laufe der Arbeit (Projekt Mitte) stellte sich aber heraus, dass die anfangs erstellte Planung zu optimistisch war. Die Komplexität verdeutlichte sich mit Fortschritt der Analyse/Spezifikation und bei Beginn der Arbeiten am Prototypen.

Auf Antrag des Studenten wurde die Arbeit bezüglich Treiberentwicklung ausgelassen und stattdessen ein Client/Server System konzipiert, welche die Funktionalität des Treibers in einer Java Applikation weit möglichst abdeckt. Dies war in Woche 04 der Fall.

In der Woche 11 wurde zudem festgestellt, dass die Entwicklung der Brokerkomponente nicht parallel mit dem des Clients durchgeführt werden kann. Auf Antrag des Studenten wurde die Entwicklung des Brokers gestoppt, bzw. diese für den Client aufgewendet.

Gegeben durch eine Einzelarbeit war die Manpower beschränkt. Nebst dem Programmieren musste viel Zeit für Infrastrukturarbeiten aufgewendet werden und verschlang die Dokumentation mit mehr als 1/3 der gesamt Projektdauer am meisten Zeit.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

Protokoll Woche 01 vom 20. Februar 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 20.02.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Telefonkonferenz		
Nächste Sitzung:	Fr 27.02.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Telefonkonferenz		

Aufgaben

- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Info E-Mail mit Liste von zu besprechenden Themen (Traktandenliste) für nächste Sitzung bis Do 26.02.2009 per E-Mail an (ad) senden
- (fv) Dokument des Projektplans in einem ersten Entwurf bis Fr 27.02.2009 per E-Mail an (ad) senden

Allgemeine Informationen

- (fv) kontaktierte (ad) per E-Mail für einen möglichen Termin für die erste Besprechung der Studienarbeit. Da (ad) erst ab März 2009 an der HSR ist, wurde eine Telefonkonferenz vereinbart.
  Telefonnummern:
  Prof. Dr. Axel Doering: +49 3641 220 255
  Florian Vetter: +41 44 485 66 95
- In dem Telefongespräch stellten sich die Projektmitarbeiter kurz vor:
  Zur Person des Dozenten:
  - arbeitete bei einem Medizingerätehersteller
  - Idee zu dieser Studienarbeit kam dem Dozenten während seiner Arbeitszeit
  - nimmt Tätigkeit ab Monat März 2009 im IFS an der HSR auf
- Folgende Erwartungen wurden Seitens (ve) für die Studienarbeit geäussert:
  - kennenlernen eines neuen Bereiches (Medizininformatik)
  - Anwenden der bisher erlernten Kenntnisse an einem realen Projekt
  - möchte TDD (Test Driven Development) einsetzt - schreiben von Systemtests
  - Einsatz von CI-Server und Buildservern
  - entwickelte Lösung soll allen zur Verfügung stellen (Open Source)
  - Einsatz von Open Source Lösungen
- Infos zum DICOM-Standard:

  - kommt aus der Radiologie
  - DICOM-Server kommt meistens in Verbindung mit einem PACS, welches in der Radiologie angesiedelt ist
- Folgende Links sind interessant zur Einarbeitung:
  http://medical.nema.org/
  http://www.dcm4che.org/
- Folgende Problematik wurde erwähnt
  - es gibt viele “alte” Geräte
  - Diese Geräte haben eine Lebensdauer von ca. 10 Jahre
    Bsp. Im Jahre 2001/2002 kamen Geräte auf den Markt, welche DICOM noch nicht implementiert haben. (Bsp. Geräte wie Kameras für zum fotografieren der Netzhaut)
  - Arzt gibt Daten von Patient in PC ein → Zugriff auf Daten sind nur von lokalem Gerät aus möglich
- Folgende Lösung für das obenbeschriebene Problem wäre denkbar:
  - Einfache Idee eines virtuellen Druckertreibers, welches auf dem Client installiert wird.
  - Via Druckfunktion kann der Endbenutzer Daten in ein DICOM senden.
  - Im Hintergrund wird eine Verbindung zu dem DICOM-Archiv aufgebaut und ein neuer DICOM-Datensatz erstellt.
  - Druckerkomponente ist unabhängig von Gerät
  - zu berücksichtigen ist:
    - Vielfalt des DICOM-Archives
    - Lösung sollte Open Source verfügbar sein.
    - Lösung muss mindestens IOD Secondary Capture, IOD Encapsulated PDF (das PDF enthält dabei gewisse Metainformationen) unterstützen.
- Folgendes Vorgehen wurde am Telefon besprochen:
  - DICOM Server aufsetzen
  - Beispielanwendung aussuchen (= emuliert Medizin Gerät, Bsp. einfacher Bildbetrachter = DICOM-Viewer)
  - Druckertreiber entwickeln (da muss auf der grünen Wiese anfangen werden) und dann in Beispielanwendung verwenden.
    - Tipp: Dinge Zusammensuchen, was es schon alles gibt.
  - Beim Druckertreiber wird der Fokus auf Windows XP Pro / Windows XP Embedded gesetzt, da ca. 70% der Anwendungen auf Windowssystemen betrieben werden.
  - Unterstützende Literatur (Buch) für Treiberprogrammierung in Windows organisieren, kaufen.
  - Programmiersprache: C++
  - Open Source Welt: Es gibt Druckertreiber, welche Pdf Dateien erzeugen
- Protokoll
  - Geräte sind **transportabel**, d.h. haben nicht immer eine aktive Netzwerkverbindung
  - Schwerpunkt auf Vertraulichkeit/Manipulationsschutz für lokaler Daten legen
  - Bilddaten werden lokal gecached (ist in einem geschützte Umfeld).
  - Kommunikation wird immer in einem geschützten Umfeld/Bereich vonstatten sein, Bsp. Kommunikation Gerät ----> DICOM (wird über Ports verbunden)

- 2-Phase Commit Protocol? Für zuverlässige Datenübertragung von Client an DICOM-Archiv, erst dann dürfen Daten lokal auf Client gelöscht werden

- Proxy für DICOM Server
  - Verteilt Konfigurationsdaten
  - kann lokal auf Gerät installiert werden
  - ist eine Proxy-Schicht zwischen Client und DICOM-Server
- Protokoll Entwurfsentscheidung
  - Der Entwickler hat volkommene Freiheit um das Protokoll zu definieren (setzt auf TCP/IP auf)
  - in den oberen Layern **XML basiert** übertragen -> Vorschlag (ad)
- Projektbasiertes arbeiten:
  - Einsatz von trac
  - diese „Umgebung“ lokal aufbauen
  - Dozent schaut dies später vor Ort an und falls sich das bewährt, wird ein trac zum Speichern/Arbeiten dieser Studienarbeit beantragt
- Vorgehensweise im Projekt:
  - agile Softwareentwicklung
  - Achtung: **Wenn entschieden, dann in Dokumentation so vermerken und konsequent über das Projekt durchziehen** (auch mit den Nachteilen, welche sich daraus ergeben)
- Folgende Dokumente möchte (ad) nach Abschluss dieser Studienarbeit:
  - Projektplan
  - Anforderungsspezifikation:
    - Software Request Specification
  - Entwurfsdokumentation:
    - Architektur
    - Designgedanken
    - Feindesign
  - Software Verifizierungsplan, d.h. Plan wie Software zu testen ist
  - Bericht – enthält Nachweis, dass ich Plan durchgeführt habe
  - Beschreibung der Build-Umgebung
    - Configurations Management Plan
  - Benutzerhandbuch (Usermanual)
    - Infos was Benutzer alles wissen muss
    - Infos zur Bedienung der Software
- Sprache der Dokumente
  - Deutsch
  - falls Englisch, gute Kenntnisse in Software Engineering (Beschreibung)

Protokoll Woche 02 vom 27. Februar 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 27.02.2009		
Beginn:	08:00 Uhr	Ende:	08:30 Uhr
Art:	Telefonkonferenz		
Nächste Sitzung:	Fr 06.03.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) funktionsfähige DICOM Infrastruktur bis Mo 02.03.2009
- (fv) Dokument Projektplanung um Teile Risikomanagement, Qualitätssicherung erweitern.
- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Dokumente (Projektplanung, Protokoll) bis Mi 04.03.2009 per E-Mail an (ad)
- (fv) Traktandenliste für nächste Sitzung bis Do 05.03.2009 per E-Mail an (ad)

Allgemeine Informationen

- (ad) wünscht, dass zukünftig Unterlagen bis spätestens Mittwoch zu gesendet werden, damit genug Zeit verbleibt die Dokumente anzuschauen.
- Dokumente mit Status versehen, z. B. „DRAFT“
- Es ist wichtig den DICOM Server schnellstmöglich installiert zu haben, um die Funktionsweise zu sehen, bzw. diesen in einer konkreten Implementierung kennen zu lernen.
- Bei der Planung ist auf die Praxis zu schauen, d.h. Frage ob Projekt in dieser Zeit realisiert werden kann, Frage ob man mögliche Alternativen zu Hilfe nehmen kann.
- (Dreieck: Funktion, Zeit, Qualität -> Zeit, Qualität ist vorgegeben, d.h. Funktion „leidet“).
- Risikomanagement beinhaltet Massnahmenplan, prüft ob 1. Szenario funktioniert, 2. gibt Vorüberlegungen, zeigen mögliche Abkürzungen auf.

Protokoll Woche 03 vom 06. März 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 06.03.2009		
Beginn:	08:00 Uhr	Ende:	08:55 Uhr
Art:	Sitzung, HSR IFS		
Nächste Sitzung:	Fr 13.03.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) Beantragen eines trac mit SVN-Repository (E-Mail an Dominik Wild IFS, dwild@hsr.ch; cc: adoering@hsr.ch)
- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Selbststudium in DICOM
- (fv) Dokument Anforderungsanalyse schreiben (Inhalt: Szenarios, UC, Sequenzdiagramm auf grober Ebene)
- (fv) Einarbeitung in Thematik Serverkomponente (zu verwendende Bibliotheken, nächste Schritte damit Lösung funktioniert)
- (fv) Dokumente (Anforderungsanalyse, Protokoll Wo3, Zeiterfassung) bis Mi 11.03.2009 per E-Mail an (ad)
- (fv) Traktandenliste für nächste Sitzung bis Do 12.03.2009 per E-Mail an (ad)

Allgemeine Informationen

- In Dokument Projektplanung Abschnitt Projektphasen Inhalt um Begründung ergänzen, wieso nicht RUP verwendet wird, dafür ein agiles Projektvorgehen.
- In Dokument Projektplanung Arbeitspakete verfeinern, d.h. Zeit für einzelne Phasen grob abschätzen (sich einen roten Faden skizzieren).
- In Dokument Projektplanung Abschnitt Zu erstellende Dokumente ergänzen, bzw. Dokumente Projektphasen zuordnen.

Protokoll Woche 04 vom 13. März 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 13.03.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Sitzung, HSR IFS		
Nächste Sitzung:	Fr 20.03.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) Selbststudium in DICOM
- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Dokument Anforderungsanalyse abschliessen (Szenario, UCs, Sequenzdiagramm auf grober Ebene, Domain-Modell)
- (fv) Dokument Anforderungsspezifikation ergänzen
- (fv) Build-Umgebung einrichten und dokumentieren
- (fv) Beispielprogramm erstellen, welche DICOM Funktionalität nutzt
- (fv) Info E-Mail bezüglich fertig erstellte Dokumente mit Link auf Repository an (ad)
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Do 19.03.2009 per E-Mail an (ad)

Allgemeine Informationen

- Ende Woche 4 soll ein „Schlussstrich“ über die Anforderungsanalyse gezogen werden.
- Die Build-Umgebung muss bis nächste Woche (Wo5) unbedingt stehen!
- Ab Woche 5 muss mit dem Prototyp begonnen werden, sonst gibt es Zeitprobleme
- Kleines Beispielprogramm erstellen, welche DICOM Funktionalität verwendet.
  - Dieses kompilieren und austesten
  - Wenn o.k., dann in Build-Umgebung integrieren.

Protokoll Woche 05 vom 20. März 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 20.03.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Sitzung, HSR IFS		
Nächste Sitzung:	Fr 27.03.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) Selbststudium in DICOM
- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Dokument Anforderungsanalyse um zusätzliche Use Cases ergänzen
- (fv) Dokument Anforderungsspezifikation ergänzen
- (fv) Beispielprogramm erstellen, welche DICOM Funktionalität nutzt
- (fv) Prototyp erstellen
- (fv) Info E-Mail bezüglich fertig erstellte Dokumente mit Link auf Repository und Zugangsdaten für svn Repository an (ad)
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Do 26.03.2009 per E-Mail an (ad)

Allgemeine Informationen

- Es wurde das Dokument Anforderungsanalyse besprochen. Dabei wurden basierend auf dem Szenario die Use Cases entsprechend angepasst, bzw. die Funktionalität in weitere Use Cases hinzugefügt.
- Es muss ein erster Prototyp diese Woche (Wo 05) erstellt werden, wo die eingesetzten Technologien verwendet. Dieser soll am nächsten Meeting besprochen werden.
- (fv) schlägt vor einen svn Repository Zugang für (ad) einzurichten - (ad) begrüsst diesen Vorschlag.

Protokoll Woche 06 vom 27. März 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 27.03.2009		
Beginn:	08:00 Uhr	Ende:	08:30 Uhr
Art:	Telefonkonferenz		
Nächste Sitzung:	Fr 03.04.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) Selbststudium in DICOM
- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Dokument Architekturdesign schreiben bis Mi 01.04.2009 Abend
- (fv) Prototyp erstellen
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Do 02.04.2009 per E-Mail an (ad)

Allgemeine Informationen

- Entwurfsphase abschliessen und in Architekturdesign-Dokument zu Papier bringen.
  - Inhalt
    - Beschreibung der Architektur
    - Formulierung der Entwicklungsumgebung
  - Info
    - Nachher können nur noch Entscheidungen in Komponenten getroffen werden
- Dokumentation Installationsanleitung
  - (ad) empfiehlt die bestehende Installationsanleitung aufzuteilen in
    - Beschreibung des Vorgehen/Einrichtung der Entwicklungsumgebung (inkl. svn Repository) (separat als Anhang zum Projektplan)
    - Beschreibung der Installation aller notwendigen Komponenten (z. B. DB für DICOM-Server, Application Server, etc.) (innerhalb Architektur/Design Dokument)
- Termin:
  - (fv) möchte wissen, ob Fr 22. Mai definitiver Abgabetermin für Studienarbeit ist, oder Fr 29. Mai wegen Frühlingsferien (unterrichtsfreie Zeit)?
    - (ad) klärt dies bis nächste Woche ab
- Vorgehen:

- (ad) findet Vorgehen soweit ok
  - Druckertreiber weglassen und Fokus auf DICOM-Kommunikation setzen
- Prototyp:
  - Instance UID -> konfigurierbar
    - Instance UID ist unabhängig von der Bit Repräsentation des DICOM Java Objekts
  - (fv) Abklären welche Ports (JMS) verwendet werden seitens Client für die Kommunikation mit dem DICOM-Broker.
    - Muss klar sein, denn medizinische IT-Infrastruktur ist bekannt für ihre Restriktionen

Protokoll Woche 07 vom 03. April 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 03.04.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Telefonkonferenz		
Nächste Sitzung:	Fr 17.04.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) Selbststudium in DICOM
- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Prototyp fertigstellen
- (fv) Info E-Mail mit Stand der Arbeiten an (ad)
- (fv) Dokumentation wie Projekt aus Repository ausgechecked und ausgeführt werde kann.
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Do 16.04.2009 per E-Mail an (ad)
- (ad) klärt ab, wann der definitive Endtermin für die SA ist

Allgemeine Informationen

- In dieser Sitzung wurde das Architektur-Design besprochen.
  - Änderungen, bzw. Ergänzungen wurden festgehalten um in Dokumentation zu ergänzen.
- Termin:
  - Da in der Woche 08-09 die Frühlingsferien/unterrichtsfreie Zeit sind (Dauer: Do 09.04.2009 – Mi 15.04.2009) wird die Studienarbeit um 1 Woche nach hinten verschoben.
  - (ad) klärt ab, ob Endtermin neu der Fr 29.05.2009 ist.
- Prototyp:
  - fertig programmieren
  - GUI

Protokoll Woche 09 vom 17. April 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 17.04.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Telefonkonferenz		
Nächste Sitzung:	Fr 24.04.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) Selbststudium in DICOM
- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Prototyp mit Funktionalität ausbauen und in Repository einchecken
- (fv) Info E-Mail mit Stand der Arbeiten an (ad) anfangs Wo 10 (Mo ist gewünscht) inkl. Link zu fertig erstelltem Doument „Architektur-Design_v2.0“ auf Repository.
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Do 23.04.2009 per E-Mail an (ad)
- (ad) liest das erstellte Dokument „Architektur-Design_v2.0“ bis zum Meeting durch und gibt (ve) dazu ein Feedback.

Allgemeine Informationen

- In dieser Sitzung wurde der Prototyp so weit als möglich besprochen. Da es Probleme mit der Integration des Prototypen in das Client Projekt gab, wurde der Client Prototyp vorübergehend als eigenständiges Projekt erstellt und lokal auf der Workstation des Entwicklers gespeichert.
- (ad) findet es wichtig, dass auch der Prototyp ins Repository eingechecked werden muss – dies soll umgehend nachgeholt werden.
- (ad) hat abgeklärt, dass der Endtermin der Studienarbeit wie gehabt Fr 22.05.2009 ist, d.h. die Studienarbeit ist dann abzugeben. Die in der Woche 08-09 als Frühlingsferien/unterrichtsfreie Zeit ausgefallene Unterrichtszeit wird ganz an die Dauer der Studienarbeit angerechnet.
- Bilder, welche an den DICOM Server gesendet werden, sind eigenständig. Durch die Identifikation mittels Instance UID können diese serverseitig referenziert werden.
- Prototyp soll mit Unit-Tests ergänzt werden.
- Dokumentation so gut wie möglich auf aktuellem Stand halten.
- Anfangen mit Programmierung der Client-Broker Komponente.

- Build-Server sollte wenn möglich eingesetzt werden.

Protokoll Woche 10 vom 24. April 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 24.04.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Sitzung, HSR IFS		
Nächste Sitzung:	Fr 01.05.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Prototyp mit Funktionalität erweitern
  - Problem mit Persistenz der Client-Daten lösen
  - Logik zum Erstellen der DICOM-Objekte und Senden an DICOM Server ausbauen
  - Funktionalität in Broker-Komponente integrieren
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Do 30.04.2009 per E-Mail an (ad)
- Dokumentation auf aktuellsten Stand bringen

Allgemeine Informationen

- In dieser Sitzung wurde der aktuelle Stand des Prototypen besprochen. Die Funktionalität des Clients erlaubt Aufträge, bzw. die dazugehörigen Ressourcen zu erfassen und abzuspeichern. Daneben können Aufträge via Suchfeld durchsucht werden, die dann entsprechend als Resultat in der Auftragstabelle angezeigt werden.
- Der Prototyp beinhaltet zudem ein Form zum Erfassen der Optionen, bzw. eine Menuoption zum Aufbauen einer Verbindung an den Broker.
- Es existiert noch ein Problem beim Persistieren der Ressource-Beschreibung – (fv) versucht dies auf nächste Woche zu lösen.
- Die Funktionalität zum erstellen der DICOM-Objekte liegt in einer eigenständigen Klasse vor und soll bis zum nächsten Mal fertig gestellt werden.
- (ad) Verlangt, dass die Stelle im Programm für Secondary Capture konfigurierbar sein soll, ohne dass der Client neu Kompiliert werden muss.
- (ad) empfiehlt (fv) auf dem HSR Wiki die Templates aus dem SE2 Projekt anzuschauen und die Gliederung auf die Dokumentation entsprechend zu übertragen.
- Dokumentation so gut wie möglich auf aktuellem Stand bringen.

Protokoll Woche 11 vom 01. Mai 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 01.05.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Sitzung, HSR IFS		
Nächste Sitzung:	Fr 08.05.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Prototyp mit Funktionalität erweitern
  - Problem mit Persistenz der Client-Daten lösen
  - Logik zum Erstellen der DICOM-Objekte und Senden an DICOM Server ausbauen
  - Funktionalität in Broker-Komponente integrieren
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Do 07.05.2009 per E-Mail an (ad)
- (fv) Dokument Architektur-Design umschreiben bis nächste Woche (Wo 12)
- (fv) Entwurf der Beschreibung der Studienarbeit bis nächste Woche (Wo 12) an (ad) zusenden
- (fv) Erstellen einer Taskliste zur übersicht der noch zu erledigenden Tasks
- (fv) Erstellen einer knappen Gliederung (Überschriften, kurzer Inhalt) für Dokument „Einstiegspunkt zur Dokumentation“
- Dokumentation auf aktuellstem Stand bringen

Allgemeine Informationen

- In der Sitzung wurden folgende Prioritäten bezüglich Realisierung definiert, da die Zeit langsam dem Ende entgegen geht:
  - Prio 1: Client muss funktionsfähig sein
    - Persistenz
    - Konfigurations Optionen
    - Validator der Benutzereingabe bei der Ressource Beschreibung
    - Funktionalität zum direkten Senden einbauen
  - Prio 2: mit Storage Commitment
  - Prio 3: Broker

- (ad) schlägt vor die Problematik mit dem ausbleibenden Storage Commitment (SC), durch den Einsatz eines amderen Client auszutesten. Dies ermöglicht die Problemeingränzung, ob SC server- oder clientseitig nicht funktioniert.
- (fv) möchte (ad) das Dokumente zur Beschreibung der Studienarbeit zusenden, damit dieses gegengelesen werden kann. (ad) begrüsst diesen Entscheid.
- (fv) schlägt vor in der letzten Woche (Wo 14) die wöchentliche Sitzung am Fr auszulassen, dafür diese vorzuziehen. Als Sitzungstermin wurde der Mi 20.05.2009 um 17:00 Uhr im IFS festgelegt, Ausweichtermin wäre Do 21.05.2009 um 07:00 Uhr
- Dokumentation
  - Installationsanleitung
    - Inhalt: wie bringe ich den Client als Software auf das Endgerät
  - Einstiegspunkt zur Dokumentation
    - Verweist auf Detaildokumente (z.B. Architektur-Design, Testplan)
    - Aufzeigen alternativer DICOM Bibliotheken
    - Persönlicher Bericht
      - Schlussfolgerungen (wo war man zu optimistisch (Broker/Druckertreiber), wo ok und am Schluss Erkenntnis daraus ziehen)
      - Prioritäten in den Anforderungen
        - Aufwandschätzung war zu optimistisch
        - Beschreibung warum Anforderungen nicht umgesetzt werden konnten
    - Zeitauswertung
    - Anforderungsanalyse
      - Anforderungen dürfen nicht gelöscht werden, wenn diese nicht innerhalb der Studienarbeitszeit realisiert werden konnten.
- Formalitäten Dokumentation
  - Abgabe am 22.05.2009 an Betreuer
  - alles ausgedruckt
  - CD-ROM mit Inhalt
- (ad) schlägt vor eine Taskliste zu erstellen, damit die noch zu erledigenden Punkte für die letzten 3 Wochen einzusehen sind.
- (ad) hat abgeklärt, dass eine Präsentation der Studienarbeit gewünscht wird. Diese ist ca. 2 Wochen nach Ende der Studienarbeit. Er wird eine E-Mail abteilungsintern versenden, falls Interessenten dazukommen möchten.

Protokoll Woche 12 vom 08. Mai 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 08.05.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Sitzung, HSR IFS		
Nächste Sitzung:	Fr 15.05.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Ri4idd2dicom-Client
  - Problem mit Persistenz der Client-Daten lösen
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Do 14.05.2009 per E-Mail an (ad)
- (fv) Dokument Architektur-Design fertigstellen bis Di 12.05.2009 Abend (Wo 13)
- (fv) Taskliste aktualisieren
- (fv) Dokument „Einstiegspunkt zur Dokumentation“ fertigstellen
- Dokumentation auf aktuellstem Stand bringen für Review (Status: Review)

Allgemeine Informationen

- Es existiert momentan ein Fehler beim Storage Commitment einer importierten DICOM-Ressource mittels dem dcm4che Toolkit - das Importieren funktioniert ohne Probleme. Wird hingegen ein Storage Commitment für diese Ressource verlangt, dann taucht folgender Fehler auf:
  Storage Commitment request failed with status: A801H
- Mittels online Nachforschungen wurde herausgefunden, dass der Status A801H golgende Bedeutung hat: A801H Refused Move destination unknown
- Dieses Fehlverhalten wurde an der Sitzung besprochen und nach mögliche Ursachen gesucht. Es ist momentan nicht geklärt wo der Fehler genau zu suchen ist.
  (ad) vermutet, dass diese Fehlermeldung mit Offline Storage zu tun haben könnte, welcher serverseitig noch nicht konfiguriert wurde.
- Für den weiteren Verlauf der Studienarbeit bedeutet dies, dass die Storage Commitment Funktionalität nicht richtig getestet werden kann.
- Es existiert weiterhin das Problem mit der Persistenz auf die Datenbanktabelle 'Ressourcebeschreibung'. (ad) vermutet, es könnte ein Anzeigeproblem sein, da keine

Fehlermeldungen beim Persistieren erscheinen –> (fv) versucht dies mittels Debugging zu lösen, bzw. den Fehler zu lokalisieren und wenn möglich zu lösen.

- (ad) möchte das Designdokument in der Wo13 durchlesen - daher muss dieses bis spätestens Di Abend fertig sein (Link an (ad) per E-Mail senden wenn fertig).
- In der nächsten Woche müssen alle Dokumente in einem ersten Entwurf vorliegen, d.h. den Status Review haben.
- Die fergien Dokumente sollen bis So 17.05.2005 Abend fertig sein -> (ad) gibt ein Feedback für Mo 18.05
- Das letzte Treffen wird am Mi 20.05.2009 um 17:00Uhr sein -> Ziel letzter Schliff an Dokumentation.

Protokoll Woche 13 vom 15. Mai 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 15.05.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Sitzung, HSR IFS		
Nächste Sitzung:	Fr 22.05.2009		
Beginn:	08:00 Uhr	Ende:	~09:00 Uhr
Je nach Bedarf	14:00 Uhr	Ende:	~17:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Do 21.05.2009 per E-Mail an (ad)
- (fv) Dokument Architektur-Design Inhaltsverzeichnis bis So 17.05.2009 per E-Mail an (ad) senden
- (fv) Dokument Architektur-Design bis Mi abschliessen und Info E-Mail mit Link an (ad) senden
- (fv) Taskliste aktualisieren
- (fv) Dokument „Einstiegspunkt zur Dokumentation“ fertig stellen
- (fv) Dokument „über DICOM“ erstellen mit spezifischem DICOM Inhalt
- Dokumentation abschliessen (Status: Final)
- (ad) sendet (fv) ein Beispiel eines guten Architektur-Design Dokumentes

Allgemeine Informationen

- Abgabetermin
  - (ad) berichtete, dass die Studienarbeit bis Ende Semester geht, d.h. es bleibt eine zusätzliche Woche um die Arbeit abzuschliessen. **Abgabetermin neu Fr 29.05.2009**
- In der Woche 13 wurde der Schwerpunkt auf die Dokumentation gesetzt
- Die Probleme mit dem Storage Commitment einer importierten DICOM-Ressource ist immer noch gegeben, bzw. das Problem mit dem Persistieren der Daten für die Ressourcen Beschreibung (siehe Protokoll Wo12)
- Im Dokument Anforderungsanalyse
  - Domainmodel
    - keine Navigierbarkeit (Pfeil löschen)
    - Attribute fehlen -> diese ergänzen
  - Systemsequenzdiagramm ist zu detailliert
    - austauschen mit demjenigen aus dem Dokument Architektur-Design

(Abschnitt: Funktionalität Client/Broker/Server)

- Schwerpunkte für nächstes Meeting
  - Feedback auf Dokument Architektur-Design
  - Dokumentation ganz anschauen
  - Termin: evtl. Fr Nachmittag freihalten für zusätzliche Besprechungsmöglichkeit

Protokoll Woche 14 vom 22. Mai 2009

Organisatorisches

Anwesende

Betreuer:	Prof. Dr. Axel Doering	(ad)
Student:	Florian Vetter	(fv)
Protokoll:	(fv)	

Termine

Sitzungsdatum:	Fr 22.05.2009		
Beginn:	08:00 Uhr	Ende:	09:00 Uhr
Art:	Sitzung, HSR IFS		
Nächste Sitzung:	Mi 27.05.2009		
Beginn:	17:15 Uhr	Ende:	~18:00 Uhr
Art:	Sitzung, HSR IFS		

Aufgaben

- (fv) kurzes Protokoll der letzten Sitzung schreiben
- (ad) Schaut ausgedruckte Dokumente an und gibt ein Feedback dazu bis anfangs Wo15
- (fv) Info E-Mail mit Link auf Traktandenliste für nächste Sitzung bis Di 26.05.2009 per E-Mail an (ad)
- (fv) Dokumentation abschliessen (Status: Final)
- (fv) Erstellen eines PDF-Dokumentes, welches die ganze Dokumentation beinhaltet

Allgemeine Informationen

- In der Woche 14 wurde der Schwerpunkt auf das Abschliessen der Dokumentation gesetzt
- (fv) hat alle erstellte Dokumente zur Sitzung ausgedruckt mitgebracht
- (ad) nimmt sich Zeit die erstellten Dokumente durchzusehen und möchte bis spätestens anfangs der Wo15 (fv) ein Feedback dazu geben, damit Zeit besteht die Dokumentation entsprechend anzupassen für die definitive Abgabe.
- (fv) fragte, ob (ad) einverstanden ist, wenn die fertige Dokumentation doppelseitig bedruckt abgegeben wird, da sonst zu viel Papier gedruckt wird – (ad) ist einverstanden mit diesem Vorschlag
- (ad) wünscht, dass in der Finalen Abgabe ein PDF-Dokument erstellt wird, welche die ganze Arbeit beinhaltet.
- Es wurden kurz alle erstellten Dokumente überflogen und nachfolgende Empfehlungen durch (ad) abgegeben:
  - Inhalt Installation Repository, Trac aus der Betreiberdokumentation ins Dokument Configurations Management Plan verschieben.
  - Im Dokument Configurations Management Plan eine Beschreibung der Struktur innerhalb des SVN-Repositorys erstellen
  - Dokument Systemtests vervollständigen mit Inhalt: Was ist zu testen (welche Anforderungen), Kriterien (was erwarte ich) und eine Beschreibung wie der Test durchgeführt wurde.

- Zustandsdiagramm im Dokument Architektur-Design UML-konform anpassen.

- Die Probleme mit dem Storage Commitment einer importierten DICOM-Ressource ist immer noch gegeben, bzw. das Problem mit dem Persistieren der Daten für die Ressourcen Beschreibung (siehe Protokoll Wo12)
  - (fv) wendet keine Zeit mehr dafür auf, da die Dokumentation höchste Priorität besitzt.