

Workflowsteuerung und Bilddarstellung in Medizinischen Informationssystemen mit mobilen Clients

Studienarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2009

Autoren: Mario Guhl und Yves Thrier
Betreuer: Prof. Dr. Axel Doering
Gegenleser: Hansjörg Huser

Erklärung

Wir erklären hiermit,

- dass wir die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt haben, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass wir sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben haben.

Rapperswil, den 17.12.2009

Mario Guhl, Unterschrift: _____

Yves Thrier, Unterschrift: _____

Danksagung

Für die Unterstützung bei unserer Studienarbeit möchten wir folgenden Personen noch einen besonderen Dank ausrichten:

Prof. Dr. Axel Doering, für seine Unterstützung und sein wertvolles und konstruktives Feedback.

Hansjörg Huser, für das kurzfristige Einspringen.

Marianne Heinzer, für das Durchlesen und Korrigieren der gesamten Arbeit und die Aufdeckung unserer grammatikalischen Verbrechen.

Christian Ramseier und Sebastian Hunkeler, für das Feedback und die Verbesserungsvorschläge des Domain-Modells.

Corsin Camichel, für das Durchführen der Usability-Tests.

Inhaltsverzeichnis

1	Abstract	1
2	Management Summary	3
3	Projektplan	5
3.1	Dokument	5
3.1.1	Dokumentinformationen	5
3.1.2	Dokumenthistory	5
3.2	Einführung	5
3.2.1	Zweck	5
3.2.2	Gültigkeitsbereich	6
3.2.3	Übersicht	6
3.3	Projekt Übersicht	6
3.3.1	Zweck und Ziel	7
3.3.2	Annahmen und Einschränkungen	7
3.4	Projektorganisation	7
3.4.1	Organisationsstruktur	8
3.4.2	Externe Schnittstellen	8
3.5	Managementabläufe	8
3.5.1	Projekt Kostenvoranschlag	8
3.5.2	Projektplan	8
3.5.2.1	Zeitplan	8
3.5.2.2	Messkriterien Zeitplan	8
3.5.2.3	Meilensteine	9
3.5.2.4	Messkriterien Meilensteine	9
3.5.2.5	Iterationsplanung	10
3.5.2.6	Besprechungen	10
3.5.2.7	Releases	10
3.6	Risikomanagement	10
3.6.1	Risiken	11
3.6.2	Massnahmen und Eintrittserkennung	11
3.6.2.1	R1 – Hardware Ausfall	11
3.6.2.2	R2 – SVN Ausfall VPN	12
3.6.2.3	R3 – Datenablage Ausfall	12
3.6.2.4	R4 – Kommunikation	12
3.6.2.5	R5 – Krankheit	12
3.6.2.6	R6 – Projektumfang	13
3.6.2.7	R7 – Technologiestudium (Kategorie 1)	13
3.6.2.8	R8 – Datenverlust	14

3.6.2.9	R9 – Infrastruktur (Kategorie 1)	14
3.6.3	Eingetretene Risiken	15
3.7	Arbeitspakete	15
3.8	Software und Infrastruktur	18
3.8.1	IDE	18
3.8.2	Text- und Dokumentationswerkzeuge	18
3.8.2.1	Allgemeine Dokumentation	18
3.8.2.2	Codedokumentation	18
3.8.3	Versionsverwaltung	18
3.8.4	Buildsystem	18
3.8.5	Bugtracking	19
3.8.6	Codereview	19
3.8.7	Geräte	20
3.8.8	Software	20
3.8.9	Zielhardware	20
3.8.9.1	Kriterien	21
3.8.9.2	Produkte	21
3.8.9.3	Vergleich	22
3.8.9.4	Auswahl	23
3.8.9.5	Produktlaufzeit	23
3.8.9.6	Investitionskosten	24
3.9	Change Management	24
3.10	Qualitäts Management	25
3.10.1	Codedokumentation	25
3.10.2	Codereview	25
3.10.3	Codierrichtlinien	25
3.10.4	Arbeitsablauf	25
3.10.4.1	Arbeitsprozess	25
3.10.4.2	Entwicklungsprozess	25
3.10.5	Versionierung	26
3.10.5.1	Branching/Tagging	26
3.10.6	Testing	26
3.10.6.1	Unit Tests	26
3.10.6.2	Usability Tests	26
3.10.6.3	Performance Test	27
3.10.6.4	Systemtest	28
3.10.6.5	User Acceptance Test	28
3.10.6.6	Qualitätsstand	28
3.10.6.7	Bugtracking	29
3.10.6.8	Metriken	29

4	Anforderungsspezifikation	31
4.1	Dokument	31
4.1.1	Dokumentinformationen	31
4.1.2	Dokumenthistory	31
4.2	Einführung	31
4.2.1	Zweck	31
4.2.2	Gültigkeitsbereich	31
4.2.3	Übersicht	32
4.3	Allgemeine Beschreibung	32
4.3.1	Produkt Perspektive	32
4.3.2	Produkt Funktion	32
4.3.3	Benutzer Charakteristik	32
4.3.4	Einschränkungen	32
4.3.5	Abhängigkeiten	32
4.4	Funktionale Anforderungen	33
4.4.1	Muss Anforderungen	33
4.4.2	Optionale Anforderungen	33
4.5	Nicht funktionale Anforderungen	33
4.5.1	Funktionalität	34
4.5.1.1	Richtigkeit	34
4.5.1.2	Interoperabilität	34
4.5.1.3	Sicherheit	34
4.5.1.4	Ordnungsmässigkeit	34
4.5.1.5	Konformität	34
4.5.2	Zuverlässigkeit	35
4.5.2.1	Fehlertoleranz	35
4.5.2.2	Robustheit	35
4.5.2.3	Wiederherstellbarkeit	35
4.5.2.4	Konformität	35
4.5.3	Benutzbarkeit	36
4.5.3.1	Verständlichkeit	36
4.5.3.2	Erlernbarkeit	36
4.5.3.3	Bedienbarkeit	36
4.5.3.4	Attraktivität	36
4.5.3.5	Konformität	36
4.5.4	Effizienz	37
4.5.4.1	Zeitverhalten	37
4.5.4.2	Verbrauchs-/Ressourcenverhalten	37
4.5.4.3	Konformität	37
4.5.5	Änderbarkeit	37
4.5.5.1	Stabilität	37
4.5.5.2	Analysierbarkeit	37

4.5.5.3	Modifizierbarkeit	37
4.5.5.4	Testbarkeit	38
4.5.5.5	Konformität	38
4.5.6	Übertragbarkeit	38
4.5.6.1	Anpassbarkeit	38
4.5.6.2	Installierbarkeit	38
4.5.6.3	Konformität	38
4.6	Schnittstellen	39
4.6.1	Benutzerschnittstelle	39
4.6.2	Hardwareschnittstelle	39
4.6.3	Softwareschnittstelle	39
4.6.4	Datenbankschnittstelle	39
4.6.5	Kommunikationsschnittstelle	39
4.7	Lizenzanforderungen	39
4.8	Verwendete Standards	40
4.9	Use Cases	41
4.9.1	Diagramm	41
4.9.2	Brief	41
4.9.2.1	UC 1: Setting up a connection to a DICOM-server	41
4.9.2.2	UC 2: View Task for all «application entities»	42
4.9.2.3	UC 3: Proceed a Task	42
4.9.2.4	UC 4: Search for patient	42
4.9.2.5	UC 5: View Patient Data	42
4.9.2.6	UC 6: View Patient Study	42
4.9.3	Aktoren	42
4.9.4	Stakeholders	42
4.9.4.1	Sicht Kunde	42
4.9.4.2	Sicht Projektteam	43
4.9.4.3	Sicht Endbenutzer	43
4.9.5	Fully Dressed	43
4.9.5.1	UC 1: Setting up a connection to a DICOM-server	43
4.9.5.2	UC 2: View Task for all «application entities»	44
4.9.5.3	UC 3: Proceed a Task	45
4.9.5.4	UC 4: Search for patient	46
4.9.5.5	UC 5: View Patient Data	47
4.9.5.6	UC 6: View Patient Study	48
5	Domain Analyse	51
5.1	Dokument	51
5.1.1	Dokumentinformationen	51
5.1.2	Dokumenthistory	51

5.2	Einführung	51
5.2.1	Zweck	51
5.2.2	Gültigkeitsbereich	52
5.2.3	Übersicht	52
5.3	DICOM-Einführung	52
5.3.1	Was ist DICOM?	52
5.3.2	Das Informationsmodell	53
5.3.3	Service-Klassen	53
5.3.4	Composite und Normalized	54
5.3.5	Verbindungsaufbau	54
5.3.6	Tags	54
5.4	Domainmodell	55
5.4.1	Strukturdiagramm	55
5.4.2	Konzeptbeschreibung	56
5.4.2.1	Patient	56
5.4.2.2	Study	56
5.4.2.3	Task	56
5.4.2.4	Surgery	56
5.4.2.5	StructuredReport	56
5.4.2.6	Serie	56
5.4.2.7	Image	56
5.4.2.8	ProcedureState	56
5.4.2.9	Modality	57
5.4.2.10	ServiceClass	57
5.4.2.11	PatientLookupServiceClassUser	57
5.4.2.12	WorklistServiceClassUser	57
5.4.2.13	NetworkConnection	57
5.5	Systemoperationen/System Sequenz Diagramme	58
5.5.1	Contract/SSD für UC1	58
5.5.1.1	Sequenzdiagramm	58
5.5.1.2	Contract: GetConnectionConfiguration	58
5.5.1.3	Contract: SetHost	58
5.5.1.4	Contract: SetPort	59
5.5.1.5	Contract: SaveConfiguration	59
5.5.2	Contract/SSD für UC2	60
5.5.2.1	Sequenzdiagramm	60
5.5.2.2	Contract: GetWorklistFor	60
5.5.3	Contract/SSD für UC3	61
5.5.3.1	Sequenzdiagramm	61
5.5.3.2	Statediagramm	61
5.5.3.3	Contract: GetNextTask	61
5.5.3.4	Contract: SetInProgress	62

	5.5.3.5	Contract: EnterResults	62
	5.5.3.6	Contract: SetFinishedTask	62
	5.5.4	Contract/SSD für UC4	63
	5.5.4.1	Sequenzdiagramm	63
	5.5.4.2	Contract: GetPatient	63
	5.5.4.3	Contract: SelectPatientBy	64
	5.5.4.4	Contract: SelectPatient	64
	5.5.5	Contract/SSD für UC5	65
	5.5.5.1	Sequenzdiagramm	65
	5.5.5.2	Contract: GetAvailablePatientInformations	65
	5.5.5.3	Contract: GetInformationBy	65
5.6		Logische-/Technologieschichten	66
	5.6.1	Technologie-Layers	66
	5.6.1.1	ihe4mobile	66
	5.6.1.2	MIDP	66
	5.6.1.3	CLDC	67
	5.6.1.4	KVM	67
5.7		GUI-Analyse	67
	5.7.1	Paper-Prototype	67
	5.7.1.1	Hauptmenü	67
	5.7.1.2	Taskliste	68
	5.7.1.3	Task Details	69
	5.7.1.4	Report bearbeiten	70
	5.7.1.5	Optionen	71
	5.7.2	User Environment Model	72
5.8		DICOM-Analyse	72
	5.8.1	Übersicht	72
	5.8.2	Operationen	73
	5.8.2.1	Association	73
	5.8.2.2	Composite Find	76
	5.8.2.3	Normalized Set	76
	5.8.3	Portierbarkeit	76
	5.8.4	Schichten	76
6		Software Architektur Dokument	77
	6.1	Dokument	77
	6.1.1	Dokumentinformationen	77
	6.1.2	Dokumenthistory	77
	6.2	Einführung	77
	6.2.1	Zweck	77
	6.2.2	Gültigkeitsbereich	77
	6.2.3	Übersicht	78

6.3	Architektonische Darstellung	78
6.3.1	SSD über alle Schichten	78
6.3.2	3-Layer Architektur	78
6.3.3	Client/(Server)	78
6.3.3.1	Datenfilterung	79
6.3.3.2	GUI-Synchronisation	79
6.4	Architektonische Ziele und Einschränkungen	79
6.4.1	Ziele	79
6.4.2	Einschränkungen	79
6.5	Logische Architektur	80
6.5.1	Übersicht	80
6.5.1.1	SSD über alle Layer	81
6.5.1.2	GUI	81
6.5.1.3	Business	82
6.5.1.4	Storage	82
6.5.2	Package GUI	83
6.5.2.1	Beschreibung des Package	83
6.5.2.2	Diagramm	83
6.5.2.3	Schnittstellen	84
6.5.2.4	Klassen	84
6.5.2.5	Abhängigkeiten von anderen Packages	86
6.5.3	Package Business	86
6.5.3.1	Beschreibung des Package	86
6.5.3.2	Diagramm	87
6.5.3.3	Schnittstellen	87
6.5.3.4	Operationen	87
6.5.3.5	Klassen	88
6.5.3.6	Abhängigkeiten von anderen Packages	89
6.5.4	Package Persistence	89
6.5.4.1	Beschreibung des Package	89
6.5.4.2	Diagramm	89
6.5.4.3	Schnittstellen	89
6.5.4.4	Klassen	90
6.5.4.5	Abhängigkeiten von anderen Packages	90
6.5.5	Package Network	90
6.5.5.1	Beschreibung des Package	90
6.5.5.2	Diagramm	91
6.5.5.3	Schnittstellen	91
6.5.5.4	Operationen	91
6.5.5.5	Transfersyntax	92
6.5.5.6	Klassen	93
6.5.5.7	Abhängigkeiten von anderen Packages	93

6.5.6	Package PDU	94
6.5.6.1	Beschreibung des Package	94
6.5.6.2	Diagramm	95
6.5.6.3	Presentation Data Value	96
6.5.6.4	Information Object Definition	96
6.5.6.5	Schnittstellen	96
6.5.6.6	Operationen	96
6.5.6.7	Klassen	97
6.5.6.8	Berechnung von Feld- und Nachrichtenlängen	104
6.5.6.9	P-Data-TF	104
6.5.6.10	Abhängigkeiten von anderen Packages	104
6.5.7	Package VR	105
6.5.7.1	Beschreibung des Package	105
6.5.7.2	Diagramm	105
6.5.7.3	Schnittstellen	105
6.5.7.4	Operationen	106
6.5.7.5	Klassen	106
6.5.7.6	Text Encoding	110
6.5.7.7	Transfersyntax Implicit Little Endian	110
6.5.7.8	Abhängigkeiten von anderen Packages	110
6.5.8	Package Parser	110
6.5.8.1	Beschreibung des Package	110
6.5.8.2	Diagramm	112
6.5.8.3	Schnittstellen	112
6.5.8.4	Klassen	113
6.5.8.5	Abhängigkeiten von anderen Packages	114
6.5.9	Package Util (Allgemein)	114
6.5.9.1	Beschreibung des Package	114
6.5.9.2	Schnittstellen	114
6.5.9.3	Subpackages	114
6.5.9.4	Klassen	114
6.5.10	Package Util (DICOM)	115
6.5.10.1	Beschreibung des Package	115
6.5.10.2	Schnittstellen	115
6.5.10.3	Klassen	115
6.6	Externe Libraries	115
6.6.1	Bouncycastle	115
6.6.1.1	Verwendete Algorithmen	115
6.7	Deployment	116
6.7.1	Files	116

6.8	Größen und Leistung	116
6.8.1	Geschwindigkeit	116
6.8.1.1	GUI	116
6.8.2	Netzwerk	117
7	Testing	119
7.1	Dokument	119
7.1.1	Dokumentinformationen	119
7.1.2	Dokumenthistory	119
7.2	Einführung	119
7.2.1	Zweck	119
7.2.2	Gültigkeitsbereich	119
7.2.3	Übersicht	119
7.3	Einleitung	120
7.3.1	Ergänzungen/Einschränkungen	120
7.4	Durchgeführte Tests	121
7.4.1	Unit Tests	121
7.4.2	Usability Tests	122
7.4.2.1	Fragebogen	122
7.4.3	Performance Tests	123
7.4.4	System Tests	124
7.4.4.1	Use Case Tests	124
7.4.4.2	Anforderungstest	124
7.5	Testauswertung	125
7.5.1	Erfüllungsgrad	125
7.5.2	Bewertung	125
7.6	Codestatistiken	125
7.7	Testen mit Emulator	126
8	Technischer Bericht	127
8.1	Dokument	127
8.1.1	Dokumentinformationen	127
8.1.2	Dokumenthistory	127
8.2	Einführung	127
8.2.1	Zweck	127
8.2.2	Gültigkeitsbereich	127
8.2.3	Übersicht	127
8.3	Einführung in die Arbeit	128
8.3.1	Problem-/Aufgabenstellung	128
8.3.1.1	Ausgeschriebene Studienarbeit	128
8.3.1.2	Konkrete Aufgabenstellung	128
8.3.1.3	Vorarbeiten	129

8.4	Lösungsskizze	129
8.5	Vorgehensweise	129
8.6	Ergebnisse	130
8.6.1	Effektives Resultat	130
8.6.2	Erfüllte Anforderungen	130
8.6.3	Offene Punkte/Probleme	131
8.6.4	Vergleich Soll-/Ist-Resultat	132
8.7	Schlussfolgerungen	132
8.7.1	Bewertung des Projekts	132
8.7.2	Bewertung Methodik und Tools	133
8.7.3	Zeitauswertung	133
8.7.3.1	Zeitauswertung Plan/Ist	134
8.7.3.2	Zeitauswertung Arbeitspakete	135
8.7.4	Lessons Learned	135
8.7.5	Rückblick	136
8.7.6	Ausblick	136
9	Persönliche Berichte	137
9.1	Dokument	137
9.1.1	Dokumentinformationen	137
9.1.2	Dokumenthistory	137
9.2	Persönlicher Bericht: Mario Guhl	137
9.2.1	Einleitung	137
9.2.2	Die Studienarbeit	137
9.2.3	Rückblick	138
9.2.4	Gelerntes	138
9.2.5	Fazit	139
9.3	Persönlicher Bericht: Yves Thrier	139
9.3.1	Einleitung	139
9.3.2	Die Studienarbeit	139
9.3.3	Rückblickend	140
9.3.4	Gelerntes	140
9.3.5	Fazit	140
	Literaturverzeichnis	141
	Glossar	145
	Abkürzungsverzeichnis	147
	Tabellenverzeichnis	149

Anhang	150
A Aufgabenstellung (Unterschrieben)	151
B Einrichtungsanleitung	157
B.1 Dokument	157
B.1.1 Dokumentinformationen	157
B.2 Einführung	157
B.2.1 Zweck	157
B.2.2 Gültigkeitsbereich	157
B.2.3 Übersicht	157
B.3 Java	157
B.4 Perl	158
B.5 IDE	158
B.5.1 eclipse	158
B.5.2 S60 SDK	158
B.5.3 Mobile Tools for Java	159
B.6 Buildscript Tools	159
B.6.1 Java WTK	160
B.6.2 Open Office	160
B.6.3 MiKTeX	160
B.7 Dokumentationswerkzeuge	160
B.7.1 PDF-XChange PDF Viewer	161
B.7.2 TeXnicCenter	161
B.8 Tortoise SVN	161
C Bedienungsanleitung	163
C.1 Dokument	163
C.1.1 Dokumentinformationen	163
C.1.2 Dokumenthistory	163
C.2 Zweck	163
C.3 Benutzung	163
C.3.1 Hauptansicht	163
C.3.2 Settings	163
C.3.3 Arbeitsschritte Suchmaske	164
C.3.4 Liste der Arbeitsschritte	164
C.3.5 Details eines Arbeitsschrittes	165
C.3.6 Details eines Patienten	166
C.3.7 Report	166
D Zeitplan	167

E	Betreuer-Meetings	171
F	Team-Meetings	179

1 Abstract

Abteilung	Informatik
Namen der Studierenden	Guhl Mario Thrier Yves
Studienjahr	HS 2009
Titel der Studienarbeit	Workflowsteuerung und Bilddarstellung in Medizinischen Informationssystemen mit Mobilien Clients
Examinator	Prof. Dr. Axel Doering
Themengebiet	Kommunikationssysteme
Projektpartner	–
Institut	Institut für Software

Kurzfassung der Studienarbeit

Das Ziel der Studienarbeit ist die Entwicklung einer Software für ein mobiles Endgerät, welches Daten von einem medizinischen Informationssystem abrufen kann und zur Kommunikation den *DICOM* Standard verwendet. Aufgrund des grossen Displays und der vollwertigen Tastatur wurde das Nokia N97 als mobiles Endgerät ausgewählt. Als konkrete Anforderungen wurden das Abrufen von Worklists (Planned Procedure Steps) sowie Patientendaten und das Abschliessen der Steps mit einem optionalen dazugehörigen «Structured Report» (*HL7*) festgelegt. Der «Structured Report» sollte ebenfalls mit der zu entwickelnden Software verfasst und zur Wiederherstellung gespeichert und verschlüsselt werden können.

Während der Analyse stellte sich heraus, dass die ursprünglich für den Einsatz geplante *dcm4che2* Library nicht eingesetzt werden kann, da *Java™ ME* nicht alle benötigten Sprachelemente unterstützt. Deshalb wurde nun als Hauptteil der Arbeit eine *Java™ ME* fähige Library entwickelt, die die Kommunikation mit einem medizinischen Informationssystem gemäss *DICOM* ermöglicht. Da der ganze Standard den Rahmen einer Studienarbeit um ein vielfaches sprengen würde, wurden nur die für die Studienarbeit benötigten Funktionalitäten implementiert. Dazu gehören Klassen für den Auf- und Abbau einer *DICOM* Verbindung, für den Worklist-Service und Schnittstellen zwischen Netzwerk- und Businessrepräsentation für die wichtigsten *DICOM Value Representations*. Ausserdem wurde das Abrufen einer Worklist umgesetzt und das Darstellen der dazugehörigen Patientendaten. Auch das Verfassen von Reports wurde umgesetzt, sowie deren automatische Wiederherstellung bei Systemausfall. Da medizinische Informationen besonders schützenswert sind, werden die Wiederherstellungsinformationen zudem verschlüsselt abgespeichert.

Die wichtigste Erkenntnis der Arbeit besteht darin, dass es möglich ist, *DICOM*-Services auf einem mobilen Gerät einzusetzen. Dies erfordert jedoch einen grossen Aufwand, da nicht auf bestehende Libraries aufgebaut werden kann.

2 Management Summary

Diplomanden	Mario Guhl, Yves Thrier
Examinator	Prof. Dr. Axel Doering
Experte	–
Themengebiet	Kommunikationssysteme
Industriepartner	–

Ausgangslage

Auch in der Medizinbranche haben sich Informationssysteme erfolgreich etabliert. Vierorts werden bereits diverse medizinische Geräte an diese Systeme angeschlossen um Messergebnisse, Bilder und Videodateien mit Patientendaten zu verknüpfen. Dies bedingt eine standardisierte Schnittstelle für die Übertragung und Darstellung, sowie Speicherung dieser Daten. *Digital Imaging and Communication in Medicine (DICOM)* fasst diese Anforderungen in einem offenen Standard zusammen. In *DICOM*¹, sind verschiedene Dienste verfügbar, wie beispielsweise das Verwalten von Patientendaten oder Abfragen von zu tätigen Arbeitsschritten durch ein medizinisches Gerät.

Vorgehen

Basierend auf einer mobilen Plattform soll ein Subset der *DICOM*-Dienste angesteuert und visualisiert werden. Als sprachliche Plattform dient Java™ *Micro Edition*. Das Ziel ist, ein Gerät für medizinische Fachpersonen zur Verfügung zu stellen, welches ihren Arbeitsalltag unterstützt, indem sie damit Informationen von einem *DICOM* Informationssystem abrufen können.

Ergebnisse

Ihe4Mobile ist eine Java™ *ME* Applikation, welche den *DICOM*-Worklist Dienst zur Verfügung stellt. Mittels diesem ist es möglich, Anfragen zu Arbeitsschritten für einen Patienten abzurufen. Da diese Abfrage über W-LAN erfolgt, können die zu erledigenden Aufgaben direkt vor Ort abgefragt und durchgeführt werden. Ein Anwender kann somit flexibel vor Ort geplante Schritte abrufen oder vorweg auf dem Weg zur Konsultation diese durchgehen.

¹ *Digital Imaging and Communication in Medicine*

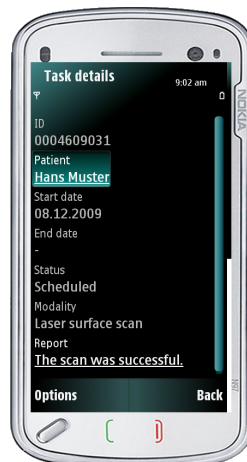


Abbildung (2.1) Beispieldetails eines Arbeitsschrittes

Ausblick

DICOM unterstützt diverse andere Dienste, wie beispielsweise das Übertragen von Bilddaten in das Informationssystem. Da nahezu jedes mobile Gerät heutzutage eine Kamera beinhaltet, wäre es denkbar, das Übertragen von Bilddateien als Erweiterung von *Ihe4Mobile* umzusetzen. Dies könnte z. B. in der Dermatologie zur Anwendung kommen, in der ein betreuender Arzt direkt Fotos von Hautstellen erstellen könnte, um diese dann in das System einzuspielen.

3 Projektplan

3.1 Dokument

3.1.1 Dokumentinformationen

Version: 1.0
Revision: Rev: 414
Status: Release
Erstellt am: 17.09.2009
Erstellt von: Mario Guhl

3.1.2 Dokumenthistory

Rev.	Datum	Wer	Änderung
0	17.09.2009	mguhl	Dokument aufgesetzt
5	22.09.2009	mguhl	Risikomanagement hinzugefügt
6	22.09.2009	ythrier	Change-/Qualitymanagement, Zeitplanung, Working Packages usw. hinzugefügt
8	23.09.2009	ythrier	Changemanagement überarbeitet, Qualitymanagement überarbeitet, Testing hinzugefügt
11	24.09.2009	ythrier	Kapitel Software überarbeitet
29	01.10.2009	mguhl	Kapitel Zielhardware hinzugefügt
31	03.10.2009	mguhl	Zusätzliche Endgeräte im Kapitel Zielhardware hinzugefügt
33	04.10.2009	mguhl	Kapitel Zielhardware fertiggestellt
64	13.10.2009	mguhl	Abschnitte Produktlaufzeit und Investitionskosten zu Kapitel Zielhardware hinzugefügt
69	14.10.2009	ythrier	Meilensteine angepasst
144	31.10.2009	mguhl	Eingetretenes Risiko eingetragen
206	12.12.2009	ythrier	Label für Testing/Riskmanagement referenzierung hinzugefügt
377	16.12.2009	ythrier	Risikomanagement Update

3.2 Einführung

3.2.1 Zweck

Dieses Dokument beschreibt den Projektplan der Studienarbeit Informatik «Workflowsteuerung und Bilddarstellung in Medizinischen Informationssystemen mit mobilen Clients» der Studierenden Mario Guhl und Yves Thrier.

3.2.2 Gültigkeitsbereich

Die hier beschriebenen Themen sind die Grundlagen des Projekts und gelten deshalb für die ganze Projektdauer.

3.2.3 Übersicht

In den nachfolgenden Sektionen werden die Thematiken des Projektplans beschrieben. Darunter fallen die Projektorganisation, Managementabläufe, Arbeitspakete und Infrastruktur. Unter anderem ist auch das Risiko- und Qualitätsmanagement ein Bestandteil dieses Dokuments.

3.3 Projekt Übersicht

Die Vernetzung von (Daten liefernden) Medizingeräten und medizinischen Informationssystemen nimmt seit einigen Jahren rasant an Bedeutung zu. Sie erlaubt die Verfügbarkeit verschiedenster Daten unabhängig vom Ort der Aufnahme ebenso wie die gezielte Steuerung des Arbeitsablaufs im komplexen Unternehmen «Arztpraxis» oder «Spital». Sie dient somit der Verbesserung der Behandlungsqualität und wird mittlerweile als der wesentlichste Faktor zur Effizienzsteigerung im Gesundheitswesen betrachtet.

Voraussetzung für eine Vernetzung von Geräten und Systemen unterschiedlichster Hersteller sind Kommunikationsstandards. *DICOM* und *HL7*¹ dienen dem Austausch von Informationen zwischen (meist bildgebenden) Medizin- und Laborgeräten und zentralen Informationssystemen (z. B. einem *KIS*² oder einem *LIS*³). In zunehmendem Masse werden Patientendaten nicht mehr an einzelnen Geräten eingetragen, sondern über einen Worklist-Mechanismus von einem führenden System abgefragt. Während in spezialisierten Fachabteilungen (z. B. Radiologie) die Auswertung von Bilddaten meist an speziellen, hochauflösenden, kalibrierten Befundstationen erfolgt, ist für viele Anwendungen (Kardiologie, Ophthalmologie, Nuklearmedizin) ein mobiler Zugriff auf Daten wichtiger als eine hohe grafische Darstellungsqualität. Dies gilt insbesondere für die Organisation des Arbeitsablaufs (welche Prozeduren sind mit welchen Patienten an welchen Geräten durchzuführen?) und die Erfassung von abrechnungsrelevanten Informationen (tatsächlich durchgeführte «procedure steps», Verbrauchsmaterial).

1 *Health Level 7*

2 *Klinik-Informationssystem*

3 *Labor-Informationssystem*

3.3.1 Zweck und Ziel

Ziel der Arbeit ist die Entwicklung eines *DICOM*-Clients für einen Pocket-PC. Damit sollen in Kommunikation mit einem *DICOM*-Server Behandlungsaufträge («Planned Procedure Steps») für verschiedene Untersuchungsstationen («Application Entities») abgefragt werden und tatsächlich durchgeführte Untersuchungen/Behandlungen («Performed Procedure Steps») zurückgemeldet werden.

Die Aufgabe kann dahingehend erweitert werden, dass auch Bilddaten vom mobilen Gerät aus abgefragt und angezeigt werden können.

Damit Geräte, welche unterschiedliche Teile des komplexen *DICOM* Standards umsetzen, zusammenarbeiten können, wurde der Metastandard *IHE*¹ entwickelt. Systeme, die bestimmte Rollen innerhalb dieses Metastandards einnehmen wollen, müssen sich ausführlichen Tests unterziehen. Diese Tests sind, z. B. für die Rollen ADT Patient Registration, Image Display oder Report Reader durchzuführen.

3.3.2 Annahmen und Einschränkungen

Pro Teammitglied rechnen wir mit einem wöchentlichen Aufwand von 18–19 Stunden. Bei Bedarf kann dieses Arbeitspensum auf 22–23 Stunden erhöht werden.

3.4 Projektorganisation

Das Projektteam besteht aus zwei Personen, wobei jede Person für spezielle Teilgebiete verantwortlich ist. Betreuer des Projekts ist Herr Axel Doering. Für die Projektarbeit, sind folgende Arbeitszeiten vorgesehen:

Dienstag, 08:10–16:50 Uhr: Arbeitszeit, Teamsitzung, weiteres Vorgehen

Freitag, 13:10–16:50 Uhr: Zusätzlich mögliche gemeinsame Arbeitszeit

Dienstag, 17:15–18:15: Wöchentliche Besprechung mit Betreuer

Mittwoch–Montag: Individuelles Arbeiten

¹ *Integrating the Health Enterprise*

3.4.1 Organisationsstruktur

Nr.	Verantwortlichkeit
Mario Guhl	Infrastruktur, Bugtracking, Latex, Architektur, <i>GUI</i>
Yves Thrier	Risikomanagement, Changemanagement, Qualitätsmanagement, Netzwerk- und Kommunikation

Tabelle (3.1) Übersicht Verantwortlichkeiten

3.4.2 Externe Schnittstellen

Betreuer des Projektes ist Prof. Dr. Axel Doering.

3.5 Managementabläufe

3.5.1 Projekt Kostenvoranschlag

Die Studienarbeit der Informatik wird mit acht *ECTS*¹ Punkten honoriert, was einem Arbeitsaufwand von 240 Stunden auf 14 Wochen entspricht. Tendenziell ist der Gesamtaufwand für dieses Projekt höher einzuschätzen als die hier veranschlagte Zeit. Dies bedeutet, dass wir durchschnittlich 19 Stunden pro Woche investieren werden. Das entspricht in der Summe 266 Stunden auf 14 Wochen verteilt. Bei Engpässen wird das wöchentliche Arbeitspensum auf maximal 23–25 Stunden erhöht.

3.5.2 Projektplan

3.5.2.1 Zeitplan

Der Zeitplan ist im Dokument «Zeitplan.xls» (siehe Anhang D auf Seite 167) zu finden.

3.5.2.2 Messkriterien Zeitplan

Um die Einhaltung des Zeitplans zu kontrollieren, werden wir folgende Messkriterien verwenden:

Teamsitzung

An den Teamsitzungen wird jedes Mitglied kurz darüber berichten, was es in der Woche an Arbeiten erledigt hat und was noch offen ist. Mithilfe des Zeitplans kann bestimmt werden, ob der Punkt abgeschlossen ist und der Zeitplan überhaupt eingehalten wurde. Dies geschieht mit dem Vergleich der noch verfügbaren Zeit zum abgeschätzten Restaufwand.

¹ *European Credit Transfer System*

Meilensteine

An den Meilensteinen existieren konkret definierte Ziele bzw. Arbeiten, welche abgeschlossen sein müssen. Es kann an diesen Punkten direkt festgestellt werden, ob der Zeitplan eingehalten wurde.

Zeiterfassung:

Die Zeiterfassung gibt Auskunft über eingesetzte Zeit und Ergebnis. Es kann so abgeschätzt werden, wie viel Aufwand noch geleistet werden muss und mit dem Zeitplan verglichen bzw. notfalls abgeglichen werden.

3.5.2.3 Meilensteine

<i>Nr.</i>	<i>Name</i>	<i>Deliverables</i>	<i>Week</i>
MS1	Projektplan	Projektplan mit Arbeitspaketplanung, Aufgabenteilung und Zeitplanung	2
MS2	Anforderungsanalyse	Anforderungsanalyse mit Katalog und ersten Use Cases (Brief und Fully Dressed)	3
MS3	Ende Elaboration	Analyse (Domainmodell, Use Case, Contracts, Activity Diagramm,...) beendet. Prototyp V1 (Worklist und Patientendaten)	6
MS4	Construction Mitte	Architekturumsetzung zu 90% fertig	10
MS5	Construction Ende	Client für Gerät fertig und getestet, Proof of Functionality and Testing erstellt SAD V1, Installationsanleitung V1, Benutzerdokumentation V1	12
MS6	Abgabe	Abstract (Sekretariat), Gesamtdokument für A. Doering	14

Tabelle (3.2) Übersicht Meilensteine

3.5.2.4 Messkriterien Meilensteine

Die Messkriterien für die einzelnen Meilensteine sind die Fertigstellung der oben genannten Deliverables.

3.5.2.5 Iterationsplanung

<i>Phase</i>	<i>No. of Weeks</i>	<i>Week</i>
Inception	2	1–2
Elaboration 1	2	3–4
Elaboration 2	2	5–6
Construction 1	2	7–8
Construction 2	2	9–10
Construction 3	2	11–12
Transition	2	13–14

Tabelle (3.3) Iterationsplanung

3.5.2.6 Besprechungen

<i>Date</i>	<i>Location</i>	<i>Person</i>	<i>Tasks</i>
Dienstag, 08:10–09:10	HSR	YT, MG	Aufgabenbesprechung, Planung, Arbeit
Dienstag, 17:15–18:15	HSR	YT, MG, AD	Arbeitsstand, Präsentation, Fragen und Rückschlüsse

Tabelle (3.4) Wöchentliche Besprechungen

3.5.2.7 Releases

Jeder Release wird im *SVN*¹ als solcher getaggt und entsprechend auch die Buildumgebung zu diesem Zeitpunkt gesichert. So kann jederzeit die Buildsituation wiederhergestellt werden.

3.6 Risikomanagement

Nachfolgend sind tabellarisch Risiken aufgelistet, welche während der Projektlaufzeit auftreten können. Die Risiken erhalten eine Einordnung bezüglich Impact, Gewichtung und Wahrscheinlichkeit sowie eine Risiko-Kategorie. Hauptrisiken befinden sich in Kategorie 1 und haben speziell ausgearbeitete Vorgänge, welche beim Auftreten durchgeführt werden müssen.

Verantwortlichkeit

Für das Erkennen von aufgetretenen Risiken ist das Projektteam verantwortlich. Pflege und Leitung des Risikomanagements fällt in die Verantwortlichkeit von Yves Thrier.

¹ *Subversion*

Dokumentation

Die Dokumentation von aufgetretenen Risiken wird unter dem entsprechenden Kapitel in dieser Sektion aufgelistet.

3.6.1 Risiken

Nr.	Name	Beschreibung	Impact	Weight	Massnahme	Prob.	Kat.
R1	Hardware Ausfall	Laptop/PC Ausfall	8h	20.00 %	R1	10.00 %	4
R2	SVN Ausfall	SVN nicht erreichbar (VPN)	10h	75.00 %	R2	65.00 %	2
R3	Datenablage Ausfall	Datenablage Ausfall	10h	75.00 %	R3	25.00 %	4
R4	Kommunikation	Netzwerkausfall	6h	40.00 %	R4	20.00 %	5
R5	Krankheit	Krankes Teammitglied	18h	30.00 %	R5	35.00 %	2
R6	Projektumfang	Zeitnot, Umfang	20h	15.00 %	R6	40.00 %	3
R7	Technologiestudium	Unterschätzt, Aufwand	15h	80.00 %	R7	55.00 %	1
R8	Datenverlust	Artefakte, etc. gelöscht	5h	40.00 %	R8	10.00 %	2
R9	Infrastruktur	Aufbau/Wartung aufw.	24h	45.00 %	R9	25.00 %	1

Tabelle (3.5) Übersicht Risiken

3.6.2 Massnahmen und Eintrittserkennung

3.6.2.1 R1 – Hardware Ausfall

Massnahme

Ein Arbeitsgerät (Workstation/persönlicher Laptop) fällt aus. Es wird auf die *HSR*¹ Infrastruktur oder ein Gerät Zuhause ausgewichen. Es kann ein Ersatzgerät innerhalb von 7 Tagen von Yves Thrier zur Verfügung gestellt werden.

Eintrittserkennung

Das Gerät startet nicht mehr oder stürzt in unregelmässigen Abständen ab. Ein effizientes Arbeiten ist kaum mehr möglich.

¹ Hochschule für Technik, Rapperswil

3.6.2.2 R2 – SVN Ausfall (VPN)

Massnahme

Auf anderen *SVN*-Server ausweichen. Nicht eingecheckte Versionen von Hand mergen um schnellstmöglich die Arbeitsumgebung wieder herzustellen. Der verwendete *SVN*-Server der *HSR* ist nur über das interne Netz oder extern via *VPN*¹-Client erreichbar. Da in der Vergangenheit bereits Probleme mit dem aktuellen *VPN*-Client aufgetreten sind, kann dies während des Projektverlaufs wieder auftreten. Sollte der Zugriff unbedingt notwendig sein, müssen die Räumlichkeiten der *HSR* aufgesucht werden, um direkt auf das interne Netz zugreifen zu können.

Eintrittserkennung

Kein Zugriff auf den *SVN* Server möglich. *VPN* Verbindung kann nicht aufgebaut werden.

3.6.2.3 R3 – Datenablage Ausfall

Massnahme

Letztes Backup der alten Ablage auf neuem Server/neuer Ablage benutzen und Differenzen zwischen lokalen Versionen und Backup manuell beseitigen, um möglichst schnell die Arbeitsumgebung wieder herzustellen.

Eintrittserkennung

Zugriff auf Datenablage über 3 Stunden lang nicht möglich.

3.6.2.4 R4 – Kommunikation

Massnahme

Daten werden vorübergehend über einen USB-Stick oder eine externe Festplatte ausgetauscht. Ein Teammitglied ist während dieser Dauer für die Konsistenz verantwortlich.

Eintrittserkennung

Es kann keine Netzwerkverbindung hergestellt werden und weder E-Mail/*SVN*/Datenablage benutzt werden.

3.6.2.5 R5 – Krankheit

Massnahme

Arbeit des kranken Teammitglieds wird während der Krankheitsphase zu 90 % vom zweiten Teammitglied übernommen. Bei länger andauernder Krankheit müssen die Projektziele

¹ *Virtual Private Network*

bzw. der Projektumfang in Rücksprache mit dem Betreuer reduziert werden, was auch eine Plananpassung zur Folge hätte.

Eintrittserkennung

Teammitglied fühlt sich schlecht (Kopfweg, Husten, Schnupfen, Hals- und Gliederschmerzen etc.). Fühlt sich nicht im Stande sein Arbeitspensum zu erfüllen.

3.6.2.6 R6 – Projektumfang

Massnahme

Wöchentliche Arbeitszeit erhöhen und mögliche Anpassungen am Projektumfang vornehmen (Review der optionalen und Muss-Anforderungen). Nach Rücksprache mit Betreuer Gesamtprojektumfang reduzieren bzw. bestimmte Anforderungen entfernen.

Eintrittserkennung

Arbeitsergebnisse werden sehr knapp vor dem angestrebten Termin (l. Zeitplan) fertiggestellt. Es wird mehr Zeit verwendet als geplant.

3.6.2.7 R7 – Technologiestudium (Kategorie 1)

Massnahme

Das Technologiestudium kann in diesem Projekt ein Genickbrecher sein. Eine Unterschätzung des Technologiestudiums hat Auswirkungen auf den Projektumfang (siehe 3.6.2.6). Sollte sich ersichtlich zeigen, dass das Technologiestudium bedeutend mehr Zeit in Anspruch nimmt als geplant, wird primär zuerst das Wochenpensum an investierten Arbeitsstunden erhöht. Sollte dies nicht ausreichen, muss über die Reduzierung/das Weglassen von Anforderungspunkten mit dem Auftraggeber gesprochen werden oder über eine Verschiebung des Abgabetermins. Andernfalls wird es nicht mehr möglich sein, die Arbeit zur rechtzeitig fertig zu stellen.

1. Zeitpensum erhöhen
2. Anforderungen/Umfang reduzieren
3. Verschiebung Abgabe

Eintrittserkennung

Es wird bedeutend mehr Zeit benötigt als im Zeitplan vorgesehen. Verständnis der Technologie steht in keinem Verhältnis zur investierten Zeit.

3.6.2.8 R8 – Datenverlust

Massnahme

Bei Datenverlust werden kurzfristig alle aktuellen Arbeiten pausiert und die letzte Ausgangslage der Daten so nah wie möglich wiederhergestellt (mittels des Backup/Restore Prozesses und lokalen Dateien). Danach werden die verlorenen Arbeiten und Artefakte identifiziert und neu aufgearbeitet. Entsprechend des dafür benötigten Aufwandes, kann es zu Engpässen kommen (siehe 3.6.2.6 auf der vorherigen Seite), wo entsprechend verfahren werden muss. Danach kann die normale Arbeit wieder aufgenommen werden.

Eintrittserkennung

Bereits erarbeitete Dokumente sind nicht mehr vorhanden/nicht mehr auffindbar.

3.6.2.9 R9 – Infrastruktur (Kategorie 1)

Massnahme

Da eine stark toolbasierte und aufwändige Infrastruktur angestrebt wird, kann der Verwaltungs- und Konfigurationsaufwand enorm viel Zeit in Anspruch nehmen. Sollte ersichtlich werden, dass zu viel Zeit in Verwaltung und Konfiguration der Infrastruktur einfließt, werden Funktionalitäten weggelassen oder vorerst nicht mehr gepflegt (z. B. *CPD*¹ in CruiseControl). Vorher jedoch steht eine Erhöhung des Zeitpensums als Lösungsvariante im Vordergrund. Als letzte Option wird eine Anpassung der Anforderungen (nach Rücksprache mit Auftraggeber/Betreuer) bzw. ein Verschieben des Abgabetermins angesehen.

1. Zeitpensum erhöhen
2. Funktionalitäten entfernen/nicht pflegen
3. Anforderungen/Umfang reduzieren
4. Verschiebung Abgabe

Eintrittserkennung

Es wird bedeutend mehr Zeit benötigt als im Zeitplan vorgesehen. Stresssituationen treten ein.

¹ *Copy-Paste-Detector*

3.6.3 Eingetretene Risiken

<i>Risiko</i>	<i>Problembeschreibung</i>	<i>Massnahmen</i>
R6	Der Einsatz der dcm4chee-Library benötigt einen enormen Portierungsaufwand auf JavaME, da viele Funktionen, welche von dcm4chee benötigt werden, von JavaME nicht unterstützt werden.	Die für das Projekt benötigte Funktionalität der dcm4chee-Library wird vom Projektteam selber implementiert, so dass es JavaME-konform ist. Da dies einen zusätzlichen Zeitaufwand benötigt, wurden in Absprache mit dem Betreuer einige Features von den Muss- in die optionale Anforderungen verlagert (siehe Kapitel 4.4 auf Seite 33).
R7	Das Technologiestudium hat sich vom Zeitaufwand her gesehen stark vergrössert, bedingt durch die Eigenimplementation der DICOM-Funktionalität	Zeitpensum erhöht und Muss-Kriterien zu optionalen Verschieben.
R9	Latex Vorlagen und Einrichten des Buildsystems nimmt mehr Zeit in Anspruch als geplant	Zeitpensum erhöht.

Tabelle (3.6) Eingetretene Risiken

3.7 Arbeitspakete

<i>Nr.</i>	<i>Name/Category</i>	<i>Task</i>	<i>Hours</i>
1	Projektmanagement		52.0
1.1	Projektplan	Zeitplanung, Meilensteine und Aufgabenbeschreibungen zusammenstellen	20.0
1.2	Verwaltung/Infrastruktur	SVN, Buildumgebung, Bug Tracking (Projektinfrastruktur)	10.0
1.3	Qualität	Definieren von Qualitätskriterien und Sicherungsmassnahmen zur kontinuierlichen Kontrolle	3.0
1.4	Review/Anpassungen	Überprüfung der potentiellen Risiken, deren Ursachen und mögliche Massnahmen	2.0
1.5	Risikomanagement	Aufzeigen der potentiellen Risiken, deren Ursachen und mögliche Massnahmen	4.0
1.6	Dokumentvorlagen	Erstellen der Vorlagen für Protokolle und Dokumentationen	13.0

weiter auf der nächsten Seite

<i>Nr.</i>	<i>Name/Category</i>	<i>Task</i>	<i>Hours</i>
2	Anforderungen		42.0
2.1	Use Case «Brief»	Use Cases im «Brief»-Format erarbeiten	10.0
2.2	Use Case «Fully Dressed»	Use Cases im «Fully Dressed»-Format erarbeiten	12.0
2.3	Supplementary Specifications	ergänzende- und nicht-funktionale Anforderungen	6.0
2.4	Anforderungskatalog	Alle Anforderungen an das Projekt	10.0
2.5	Review Anforderungen	Abgleich Anforderungskatalog und Anforderungen nach Reviewmeeting	4.0
3	Analyse		34.0
3.1	Konzeptuelles Modell	Konzeptuelles Modell mit notwendigen Domänenklassen	4.0
3.2	Domänenmodell	Domänenmodell auf Grundlage Use Cases und Anforderungen/Spezifikationen	5.0
3.3	Contracts	Blackbox Systemcalls Contracts	6.0
3.4	System Sequence Diagramm	Sequenzdiagramme von essentiell wichtigen Programmläufen	4.0
3.5	State/Activity Diagramm	Status und Aktivitäten nicht trivialer Entitäten	2.0
3.6	Externes Design	GUI Paperprototypes, Handskizzen	7.0
3.7	Review	Ergänzungen und Aktualisierungen	6.0
4	Internes Design		53.0
4.1	Klassendiagramm	Erstellen eines umfassenden Klassendiagramms	16.0
4.2	Kommunikationsarchitektur	Festlegung und Konzeption einer internen Kommunikationsarchitektur (C/S, Events, States etc.)	12.0
4.3	Internes GUI Design	Organisation der internen Strukturen und Einstellungen/Einstellungsverwaltung)	4.0
4.4	Logische Architektur (Schichtenmodell)	Schichtenmodell, Schichtentrennung und Abstrahierung/Festlegen der Zuordnungen	7.0
4.5	Problem Domain	Konzeption der Problem Domain	8.0
4.6	Review	Konsistenzüberprüfung der Problem Domain mit Klassendiagramm, GUI Design, SSD, AD/SD etc.	6.0

weiter auf der nächsten Seite

<i>Nr.</i>	<i>Name/Category</i>	<i>Task</i>	<i>Hours</i>
5	Internes Design		131.0
5.1	Prototypen	Erstellen eines lauffähigen Prototypen anhand Problem Domain	24.0
5.2	Automation	Projektautomation (Ant)	5.0
5.3	Datenhaltung	Persistenzebene (Benutzerdaten/-einstellungen)	3.0
5.4	Fehlerbehandlung	Fehlerbehandlung/Fehlertoleranz	7.0
5.5	Review und Refactoring	Codereview und Refactoring, CruiseControl Metrikauswertungen einfließen lassen	20.0
5.6	Projekt- und Codestruktur	Strukturelle Anpassungen und Überprüfungen	7.0
5.7	Ausbau <i>GUI</i>	User Interface Gestaltung, Menü-Führung etc.	24.0
5.8	Ausbau Problem Domain	Ausbau der Problem Domäne (Business Logic, Kommunikation mit Server)	41.0
6	Test		55.0
6.1	Unit Tests	Gesamt - Testabdeckung sicherstellen.	24.0
6.2	Usability Tests	Theoretische Usability - Analyse mit Drittpersonen	8.5
6.3	Performance Tests	Performance auf Zielplattform überprüfen und allenfalls Anpassungen vornehmen	4.5
6.4	Fehlerbehebung	Bugfixing und organisatorische/berreinigende Aufgaben	10.0
6.5	System Tests	Situationstests (Use Cases und Anforderungen)	8.0
7	Dokumentation		84.0
7.1	Anleitung	Benutzerhandbuch, Installationsanleitung (Endprodukt)	10.0
7.2	Software Architektur Dokument	Zusammenstellen sämtlicher Artefakte aus Analyse und Implementation inklusive Visualisierung und Erklärung der Architektur	48.0
7.3	Abschluss	Genereller Projektabschluss, «Reserve»	12.0
7.4	Abschlussbericht	Rückblickender Bericht Projektbezogen	10.0
7.5	Erfahrungsreport	Rückblickender Bericht Selbstbezogen	4.0
8	Technologiestudium		44.0
8.1	<i>DICOM</i>	Studium des <i>DICOM</i> Standards	44.0
9	Sitzungen		39.0
9.1	Sitzungsprotokoll	Erstellen der Sitzungsprotokolle/Durchführen der Sitzung	13.0
9.2	Besprechung mit Betreuer	Wöchentliches Meeting mit Betreuer	26.0
10	Qualitätssicherung (ergänzend)		22.0
10.1	Reviews (Dokumentation/Code-Style)	Generelle Review-Zeit. Kann auch als Backup betrachtet werden	22.0

Tabelle (3.7) Arbeitspakete

3.8 Software und Infrastruktur

3.8.1 IDE

Als IDE¹ wird eclipse Galileo (Version 3.5) verwendet.

3.8.2 Text- und Dokumentationswerkzeuge

3.8.2.1 Allgemeine Dokumentation

Da in früheren Projekten bereits gute Erfahrungen mit L^AT_EX gemacht wurden und es besser für grössere Dokumente ausgelegt ist als Word, haben wir uns entschieden, es für unsere Arbeit zu verwenden.

3.8.2.2 Codedokumentation

Da das Projekt in Java programmiert werden soll, wird für die Codedokumentation *Java Code Documentation* verwendet.

3.8.3 Versionsverwaltung

Als Versionsverwaltungssystem wird *Subversion* verwendet. Für eclipse wird das Subclipse Plugin verwendet. Für die Integration ins Dateisystem wird *TortoiseSVN* verwendet.

3.8.4 Buildsystem

Für eine Continuous-Integration benötigen wir einen Build-Server. An diesen werden folgende Anforderungen gestellt:

- Kostenlose Verfügbarkeit für Studenten
- Zugriff auf *SVN* oder *CVS*² Repository
- Unterstützung für Ant-Scripts
- Metrikanalyse
- Nice to have: *Pre-testet commit*

¹ *Integrierte Entwicklungsumgebung (engl. integrated development environment)*

² *Concurrent Versions System*

<i>Build-Server</i>	<i>SVN/CVS</i>	<i>Ant</i>	<i>Metriken</i>	<i>Pre-tested commit</i>
TeamCity	+	+	- (über Scripts möglich)	+
CruiseControl	+	+	+	-
Hudson	+	+ (easyAnt)	+	-

Tabelle (3.8) Übersicht Buildsysteme

Wir haben uns für CruiseControl entschieden, da es direkt integrierte Metrikauswertungen mit Verlauf und Veränderung über die ganze Projektphase beinhaltet. Ausserdem sind diverse weitere nützliche Plugins vorhanden wie *JCSC*¹, *CPD* und Dependency-Finder, was im Gesamten betrachtet einen bedeutenden Mehrnutzen bringt.

3.8.5 Bugtracking

An das Bugtracking werden folgende Anforderungen gestellt:

- Vollständige Unterstützung der üblichen Ticketstatus wie New, Assigned, Reopened, Resolved, Verified, Closed
- Tickethistory
- Filterfunktionen zur Auflistung von Tickets

<i>Bugtracker</i>	<i>Ticketstatus</i>	<i>Tickethistory</i>	<i>Filterfunktionen</i>
Bugzilla	+	+	+
Trac	-	+	+
Mantis	+	+	+

Tabelle (3.9) Übersicht Bugtracker

Wir haben uns für Bugzilla entschieden, da man Bugzilla mit Hilfe von Mylin in eclipse integrieren kann und damit einfach zwischen verschiedenen Bugs und Tasks hin- und her wechseln kann. Dies ist sehr hilfreich für ein einfaches und schnelles/effizientes Arbeiten, da sich das Bugtracking so direkt in die «Tool-Chain» integriert.

3.8.6 Codereview

Zur Vereinfachung von Codereviews wird *Codestriker* verwendet.

¹ *Java Code Style Checker*

3.8.7 Geräte

Als Arbeitsgeräte dienen die Arbeitsstationen im Laborraum und die persönlichen Laptops.

3.8.8 Software

Betriebssysteme (Entwicklung):

- Ubuntu 9.04
- Windows XP SP3

(Programmier-)Sprache:

- Java 6 (1.6)

Tools/*IDE*/Diverses:

- Eclipse Galileo 3.5
- Subclipse 1.6.5
- TortoiseSVN 1.6.5
- Cruise Control 2.8.1
- EclEmma Code Coverage 1.4.2
- Mylin 3.4
- Bugzilla 3.4.2
- Codestriker 1.9.9
- Metrics 1.3.6
- MiKTeX 2.8
- TeXnicCenter 1.0 C1

Tools/Programme und Werkzeuge entsprechend Studienarbeit von F. Vetter für *DICOM*-Server

3.8.9 Zielhardware

Die zu entwickelnde Software dieser Semesterarbeit «Workflowsteuerung und Bilddarstellung in Medizinischen Informationssystemen mit Mobilien Clients» hat – wie der Name bereits sagt – als Zielplattform ein mobiles Gerät. Nachfolgend werden konkrete Zielplattformen anhand von aufgestellten Kriterien verglichen und ein Zielgerät ausgewählt.

3.8.9.1 Kriterien

Folgende Kriterien sind für die Bewertung der Geräte herangezogen worden:

- Preis unter CHF 1'000
- Passt in eine Kitteltasche (Masse max. $120 \times 60 \times 20$ mm)
- Java Unterstützung
- WLAN Unterstützung
- Emulator für Entwicklung
- Mindestens 3.5 " Bildschirmgrösse
- Mindestens 128 MB RAM
- Mindestens 400 MB Festplattenspeicher
- Tastatur
- Gewicht maximal 300 g
- Akkulaufzeit (Online-Zeit bzw. Sprechzeit minimum 4 h)

3.8.9.2 Produkte

Da Mobiltelefone immer mehr Organizerfunktionen aufweisen, ist die Auswahl an entsprechenden Geräten mit reinen Organizerfunktionen stark zurückgegangen. Eine grössere Auswahl an Geräten gibt es nur noch von HP, wenige Geräte von Palm oder Acer z. B. bei [ARP09], [dig09] oder [Top09]. Aus diesem Grund wurde die Suche auf Smartphones ausgeweitet. Diese bieten zwar zusätzliche, nicht unbedingt notwendige Funktionalitäten, wären aber einfach für andere Dienste/Applikationen brauchbar; dies liegt aber ausserhalb des Kontextes dieser Arbeit. Folgende Produkte wurden näher betrachtet:

- Acer F900 (Smartphone)
- HP iPAQ 114 Classic Handheld (Handheld)
- HP iPAQ Data Messenger (Smartphone)
- HTC Dream G1 (Smartphone)
- Nokia N97 (Smartphone)
- Palm Treo Pro (Handheld)
- Samsung C6625 (Smartphone)
- Sony Ericsson XPERIA X1 (Smartphone)

3.8.9.3 Vergleich

Gerät	Preis	Masse (L/W/H)	JavaME	WLAN	Emulator	Display	RAM	Disk	Keyboard	Gewicht	Akku
F900	CHF 648.90 ^a	118/64/13 mm	(+) ^b	802.11b/g	(+) ^d	3.8 "	128 MB SDRAM	256 MB flash ^e	Touchscreen	150.0 g	6.0 h
iPAQ 114	CHF 366.00 ^f	117/69/14 mm	(+) ^c	802.11b/g	(+) ^d	3.5 "	64 MB SDRAM	256 MB flash ^e	Touchscreen ^g	114.6 g	k. A. ^h
iPAQ Data	CHF 734.00 ^f	114/57/18 mm	(+) ^c	802.11b/g	(+) ^d	2.8 "	128 MB SDRAM	256 MB flash ^e	QWERTZ	160.0 g	k. A. ^h
Dream G1	CHF 398.00 ⁱ	117/56/17 mm	+	802.11b/g	+ ^j	3.2 "	192 MB RAM	256 MB ROM ^e	QWERTZ	158.0 g	5.8 h
N97	CHF 699.00 ⁱ	118/56/19 mm	+	802.11b/g	+ ^k	3.5 "	128 MB SDRAM	32 GB ^e	QWERTZ	150.0 g	6.0 h
Treo Pro	CHF 521.00 ^f	114/60/14 mm	(+) ^c	802.11b/g	(+) ^d	k. A. ^l	128 MB RAM	100 MB ^e	QWERTZ	133.0 g	5.0 h
C6625	CHF 316.00 ⁱ	114/63/12 mm	+	-	+ ^m	2.6 "	k. A.	40 MB ^e	QWERTZ	k. A.	8.7 h
XPERIA X1	CHF 587.00 ⁱ	110/53/17 mm	+	802.11 b/g	(+) ⁿ	3.0 "	256 MB RAM	512 MB ROM	QWERTZ	145.0 g	10.0 h

a bei [web09]

b wird von Windows Mobile® unterstützt, aber nicht offiziell vom Hersteller: «Einen einheitlichen Standard für die unterstützte Funktionalität gibt es allerdings nicht und es ist auch nicht garantiert, dass auf einem bestimmten Windows-Mobile-Gerät überhaupt eine Java VM^c zur Verfügung steht.» [Pre09]

d Microsoft Device Emulator [Mic09] (nur für VS2008)

e erweiterbar mit Speicherkarte

f bei [ARP09]

g es kann per Bluetooth eine aufklappbare Tastatur angeschlossen werden

h Hersteller macht keine Angaben über die Akkulaufzeit

i bei [dig09]

j Android Emulator [And09a]

k siehe [Nok09]

l Auflösung: 320 × 320

m Samsung Mobile Innovator [Sam09]

n Sony Ericsson SDK [Son09] (nur für Visual Studio)

Tabelle (3.10) Vergleich Zielplattformen

3.8.9.4 Auswahl

Da alle reinen Pocket-PCs mit Windows Mobile® ausgestattet sind, aber die Hersteller keine konkreten Angaben machen ob Java unterstützt wird, fiel kein Pocket-PC in die engere Auswahl. Als optimale Entwicklungssysteme kommen das Nokia N97 (Symbian S60) und das HTC Dream G1 (Android) in Frage, da sie die gestellten Anforderungen am Besten erfüllen. Ausserdem bieten beide eine sehr gute eclipse Integration an und sehr gute Dokumentationen zu den entsprechenden Entwicklungstools. Für das HDC Dream G1 gibt es zusätzlich noch eine spezielle Developer Version (Android Dev Phone), welche auch benötigt wird, da man bei dieser Version keine Sicherheitseinschränkungen hat beim Aufspielen von unsignierter Software. Für die beiden Smartphones gibt es folgende Angebote:

Gerät	Lieferant	Preis
Android Dev Phone 1	Android Marked ^a	USD 399.00 (ca. CHF 415.00) ^b
HTC Dream G1 ^c	digitec ^d	398.00 CHF
Nokia N97	digitec ^d	699.00 CHF

a [And09b]

b Damit man das Produkt kaufen kann, muss man sich im Android Marked als Entwickler registrieren. Dabei fällt eine Registrierungsgebühr von USD 25.00 (ca. CHF 26.00) an.

c nicht für den Entwicklungsprozess geeignet

d [dig09]

Tabelle (3.11) Angebote für Zielhardware

3.8.9.5 Produktlaufzeit

Auf den Herstellerseiten sind keine Informationen zu finden über die Produktlaufzeit der Geräte. Vor allem bei Smartphones dürfte aber die Produktlaufzeit eines bestimmten Gerätes nicht mehr als 2–3 Jahre betragen, da die Entwicklung dort sehr schnell vorangeht. Bei erfolgreichen Modellen kann es aber durchaus sein, dass sie nach 3 Jahren noch erhältlich sind. Der Vorteil von Java aber ist ja, dass es plattformunabhängig ist. Somit kann die Applikation einfach auf ein anderes Gerät portiert werden, wenn es nicht mehr erhältlich ist. Zudem ist Symbian ein sehr erfolgreiches Betriebssystem, dass seit vielen Jahren eingesetzt wird. Und solange sich das Betriebssystem bei unterschiedlichen Modellen nicht unterscheidet, sollte es keine Probleme geben, die Software auf einem anderen Modell zu betreiben, zumal ja der Simulator bei allen Modellen desselben Betriebssystems derselbe ist. Bei den Betriebssystemen ist die Lebensdauer wesentlich höher, z. B. ist Symbian OS 9 schon seit 5 Jahren auf dem Markt [Wik09c]. Bei Android gibt es ebenfalls einen einzigen Simulator und nicht für jedes Modell einen, womit die Portierung auf ein anderes Modell ebenfalls kein Problem sein sollte. Da Android aber noch ein «junges» Betriebssystem ist, ist es schwierig abzuschätzen wie der Lebenszyklus dieses Betriebssystems ist. Falls es aber mangels Erfolg nicht verworfen wird, wird die Lebensdauer wohl ähnlich wie bei

anderen Handy-Betriebssystemen sein.

3.8.9.6 Investitionskosten

Von den Investitionskosten her gesehen, müsste man sich eindeutig für das HTC Dream G1 entscheiden, da die Anschaffungskosten für das Endgerät CHF 300.00 billiger sind als das beim Nokia N97, dies sind über 40 %. Da fallen die höheren Anschaffungskosten für das Entwicklungsmodell nicht mehr gross ins Gewicht. Man muss allerdings auch beachten, dass Android wie gesagt ein noch sehr «junges» Betriebssystem ist und noch nicht sicher abzusehen ist, ob es so erfolgreich ist, dass es auch in 5 Jahren noch verwendet wird. Wenn dies nämlich nicht der Fall wäre, würden sich die niedrigeren Anschaffungskosten schnell aufheben durch Investitionen für die Portierung auf ein anderes System. Bei dem Nokia N97 ist diese Gefahr allerdings sehr klein, da Nokia eine sehr starke Marke ist und Symbian ein sehr etabliertes Betriebssystem, das von vielen Herstellern in vielen Modellen eingesetzt wird.

3.9 Change Management

Das Change Management befasst sich mit Änderungen am Projekt oder neu auftretenden Anforderungen (während Construction oder bereits Ende Elaboration 2). Diese werden tabellarisch im Anforderungskatalog erfasst und an den Teamsitzungen diskutiert, ob sie aufgenommen werden, wie sie umgesetzt werden und wie dies in den Zeitplan integriert wird. Dies kann/wird auch unter Rücksprache mit dem Betreuer geschehen.

Erfassungskategorien:

- Notwendig
- Sinnvoll
- Angenommen
- Abgelehnt
- Implementiert

Einzelne Kategorien können sich überschneiden.

Arbeitsschritte:

1. Auftreten der neuen Anforderung
2. Einpflegen in Anforderungskatalog
3. An Teamsitzung über Vorgehen entscheiden
4. Allenfalls Anpassung Zeitplan und Anforderungskategorie

3.10 Qualitäts Management

3.10.1 Codedokumentation

Die Dokumentation des Codes geschieht in den Sourcecode Dateien und wird nach der *Java Code Documentation Notation* geführt und entsprechend auch am Ende als *HTML*¹ Dokumentation zur Verfügung stehen.

3.10.2 Codereview

Codereviews geschehen wöchentlich während der Construction unter Beihilfe von *Codestricker*. Die Einhaltung der Codier- und Dokumentationsrichtlinien werden überprüft.

3.10.3 Codierrichtlinien

Als Codierrichtlinien gelten dispositiv die Java Code Conventions [Sun99]. Folgende allgemeine Regelungen sind ebenfalls zu beachten:

- Klassennamen beginnen mit Grossbuchstaben
- Packagenamen beginnen mit Kleinbuchstaben
- Subpackages werden mit einem «.» (Punkt) voneinander abgetrennt
- Methodennamen werden klein geschrieben
- Konstanten werden komplett gross geschrieben, mit Unterstrichen als Worttrenner
- Feldinitialisierung geschieht direkt an der Definitionsstelle, sofern nicht durch Konstruktorparameter und/oder Programmlogik anders notwendig

3.10.4 Arbeitsablauf

3.10.4.1 Arbeitsprozess

Als Arbeitsprozess dient der *RUP*². Bezüglich Teamsitzungen u. ä. (siehe 3.5 auf Seite 8)

3.10.4.2 Entwicklungsprozess

Als Entwicklungsprozess wird ein «Test Driven Development» angestrebt.

¹ *Hypertext Markup Language*

² *Rational Unified Process*

3.10.5 Versionierung

Die Software- und Dokumentversionierung geschieht mittels der Versionsmanagement-Software *Subversion*. Es wird der *SVN*-Server der *HSR* verwendet, somit wird das Backup vom *HSR* Informatikdienst übernommen.

3.10.5.1 Branching/Tagging

Das Branching geschieht jeweils gleichzeitig mit dem Tagging des Releases. Fehlerbehebungen dieses Releases werden auf dem zugehörigen Branch ausgeführt. Die Entwicklung läuft parallel auf dem Trunk weiter. Die Branches werden jeweils nach den Bugfixes zusammengeführt. Releases sind jeweils am Ende jeder Construction Phase.

3.10.6 Testing

3.10.6.1 Unit Tests

Der angestrebte Entwicklungsprozess ermöglicht von Haus aus eine hohe Testabdeckung. Um diese aber kontinuierlich zu überprüfen wird lokal während der Entwicklung das Eclipse Plugin *EclEmma* verwendet. Auf Seite des Buildservers wird dies mittels des *CruiseControl* Plugins im regelmässigen Build direkt mitintegriert. Die angestrebte Unit Test-Abdeckung liegt hierbei bei 100 % oder höher.

3.10.6.2 Usability Tests

Aufgrund der Zielbranche und mangels eines Industriepartners ist es nur mit sehr grossem Aufwand möglich, einen effektiven Endbenutzer der Software zu finden. Folglich werden wir uns lediglich auf Tests bezüglich Navigierbarkeit und allgemeines Programmhandling beschränken, ohne auf die medizinspezifischen Punkte genauer einzugehen. Dies genauer (zusammen mit einem Industriepartner) auszuarbeiten würde sich als Folgearbeit auf diese Studienarbeit anbieten.

Zeitplanung

Es wird gesamthaft drei Usability Tests, jeweils am Ende jeder Construction Phase geben.

Nr.	Phase/Week	Tests
U1	Construction 1/Week 2	Navigierbarkeit, Verständlichkeit
U2	Construction 2/Week 2	Allgemeine Beschriftungen, Komplexität
U3	Construction 3/Week 2	Gesamthafter Test, Endbewertung

Tabelle (3.12) Usability Tests

Testform

Die Tests stellen sich in Form von Fragebögen dar, welche die Errechnung eines prozentualen Erfüllungsgrades ermöglichen.

Testpersonen

- Mario Guhl (Teammitglied)
- Yves Thrier (Teammitglied)
- Stefan Conzett (Externe Testperson, Geomatikstudent)
- Raphael Blülle (Externe Testperson, Medizinstudent)

Testablauf

Die Testpersonen erhalten die aktuelle Releaseversion mit einer Installationsanleitung sowie dem entsprechenden Fragebogen. Sie führen die Tests an einem Stück durch und retournieren den Fragebogen an das Projektteam. Zeitbudget nach Ausgabedatum beträgt vier Tage.

3.10.6.3 Performance Test

Ein Performance-Test wird lediglich in kleinem Rahmen durchgeführt, da nicht davon ausgegangen werden muss, dass das Endgerät spezielle Performance-Engpässe haben wird. Es wird lediglich darauf geachtet, dass die Ressourcenauslastung gering bleibt (harte Grenze) und ein flüssiges Arbeiten möglich ist.

Zeitplanung

Es wird gesamthaft zwei Performance-Tests, jeweils am Ende der zweiten und dritten Construction Phase, geben.

<i>Nr.</i>	<i>Phase/Week</i>	<i>Tests</i>
P1	Construction 1/Week 2	Flüssiges Arbeiten, Ressourcenauslastung
P2	Construction 2/Week 2	Flüssiges Arbeiten, Ressourcenauslastung

Tabelle (3.13) Performance Tests

Testpersonen

- Mario Guhl (Teammitglied)
- Yves Thrier (Teammitglied)

Die Tests werden anhand einer Checkliste durchgeführt und ebenfalls einen prozentualen Erfüllungsgrad als Kenngrösse bieten.

Testablauf

Der aktuelle Release wird auf dem Zielsystem installiert. Dieses wurde im Voraus mit einem neuen Image «gesäubert». Danach werden anhand der Checkliste die Tests durchgeführt.

Harte Grenzen

- 128 MB Hauptspeicher
- 400 MB Festplattenspeicher

Die Werte werden anhand des Betriebssystem-Monitorings ausgelesen. Diese Liste wird im Verlauf des Projekts um weitere Punkte ergänzt (Minimalanforderungen, Netzwerkauslastung etc.), jedoch kann momentan keine genauere Aussage gemacht werden.

3.10.6.4 Systemtest

Der Systemtest wird in Kombination mit dem Performance-Test auf der Zielplattform ausgeführt, jedoch separat als Kenngrösse behandelt. Zeitplanung und Testpersonen sind gleich.

Testablauf

Es wird ein Fragebogen/eine Checkliste durchzuarbeiten sein, der/die eine Kenngrösse bezüglich Erfüllungsgrad errechnen lässt.

3.10.6.5 User Acceptance Test

Mangels eines effektiven Endkunden wird auf den User Acceptance Test verzichtet. Gegebenenfalls kann dieser natürlich eingeführt werden, sollte sich industrielles Interesse abzeichnen.

3.10.6.6 Qualitätsstand

Der Qualitätsstand bezieht sich auf die diversen Testing Kapitel. Der durchschnittliche Erfüllungsgrad über Unit-, Usability-, Performance-, System- und User Acceptance Test darf 95 % nicht unterschreiten.

- >95 %: Keine Aktivität notwendig
- 90-95 %: Kleine Verbesserungen notwendig
- 80-90 %: Diverse Verbesserungen notwendig
- <80 %: Immense Verbesserungen notwendig

Die Verbesserungen und Anpassungen werden anhand der Testauswertungen mit den Testpersonen eruiert.

3.10.6.7 Bugtracking

Als Bugtrackingsystem wird Bugzilla verwendet. Ein Bugeintrag muss erstellt werden, wenn eine der folgenden Bedingungen eintritt.

Bug: Ein Entwickler entdeckt ein Bug oder eine Unschönheit im Code, für den/die er im Moment keine Zeit hat ihn/sie zu beheben.

Task: Ein Entwickler bemerkt, dass eine Funktion vergessen wurde oder die Verbesserung einer bestehenden Funktion seiner Meinung nach einen grossen Mehrwert bringt.

Grösserer Bug: Falls ein Bug oder eine Korrektur (auch Refactoring) eine Änderung von mehr als 15 Codezeilen beinhaltet, so muss dafür ein Eintrag erstellt werden, auch wenn dies sofort gelöst werden kann.

Review: Werden in einem Review Mängel oder vergessene Funktionen festgestellt, welche nicht während des Reviews korrigiert wurden, muss dafür ein Eintrag erstellt werden.

Sitzungen: Für alle in den Sitzungen beschlossenen Tasks (z. B. weiteres Vorgehen) muss ein Eintrag erstellt werden.

Status

Bugs bzw. Tasks können folgende Status haben:

New: Einträge, denen noch niemand zugewiesen wurde, der dafür verantwortlich ist.

Open: Einträge, welche schon jemandem zugewiesen sind, aber noch nicht gelöst wurden.

Assigned: Einträge, welche jemanden zugewiesen und von dem auch akzeptiert wurden.

Resolved: Einträge, welche abgearbeitet, aber noch nicht kontrolliert wurden, ob das Problem nun behoben ist bzw. das Feature implementiert wurde.

Closed: Einträge, welche abgearbeitet und auf ihre korrekte Abarbeitung überprüft wurden.

3.10.6.8 Metriken

Allgemeine Metriken

Es werden die «per-build» Metriken von CruiseControl verwendet. Für lokales Arbeiten wird auf das eclipse Plugin «Metrics» zurückgegriffen. Vor jedem Check-In wird überprüft, ob alle Kenngrössen im «grünen» Bereich sind. Eine genaue Auflistung aller Metriken und Grenzwerte wird zu einem späteren Zeitpunkt noch ergänzt.

Diverse (spez.) Metriken

Als zusätzliche Tools/Plugins werden auf dem Buildserver der *JCSC*, der *CPD* sowie der Dependency Finder eingesetzt.

4 Anforderungsspezifikation

4.1 Dokument

4.1.1 Dokumentinformationen

Version: 1.0
Revision: Rev: 416
Status: Release
Erstellt am: 24.09.2009
Erstellt von: Yves Thrier

4.1.2 Dokumenthistory

Rev.	Datum	Wer	Änderung
0	24.09.2009	ythrier	Dokument aufgesetzt
18	29.09.2009	mguhl	Funktionale Anforderungen hinzugefügt
24	29.09.2009	ythrier	Nicht funktionale Anforderungen hinzugefügt
26	30.09.2009	ythrier	Zusätzliche nicht funktionale Anforderungen hinzugefügt
27	30.09.2009	mguhl	Brief Use Cases, Aktoren und Stakeholders hinzugefügt
30	01.10.2009	ythrier	Nicht funktionale Anforderungen überarbeitet
31	03.10.2009	mguhl	Optionale funktionale Anforderungen überarbeitet
32	04.10.2009	ythrier	Nicht funktionale Anforderungen fertiggestellt
34	04.10.2009	mguhl	Alle Use Cases fully dressed erstellt
66	13.10.2009	mguhl	Use Cases überarbeitet
68	14.10.2009	ythrier	Nicht funktionale Anforderungen überarbeitet und Testkategorien hinzugefügt
69	14.10.2009	ythrier	Deliverables überarbeitet
141	31.10.2009	mguhl	Kapitel funktionale Anforderung angepasst gemäss Betreuer-Meeting vom 27.10.2009, UC1 überarbeitet, «procedure step» in UCs durch <i>Task</i> ersetzt

4.2 Einführung

4.2.1 Zweck

Dieses Dokument dient der Beschreibung der Anforderungen.

4.2.2 Gültigkeitsbereich

Dieses Dokument gilt für die ganze Projektdauer.

4.2.3 Übersicht

In den Anforderungsspezifikationen werden die funktionalen sowie die nicht funktionalen Anforderungen des Projekts beschrieben. Die funktionalen werden mit den Use Cases abgedeckt und die nicht funktionalen mit den Beschreibungen der Schnittstellen.

4.3 Allgemeine Beschreibung

4.3.1 Produkt Perspektive

Um die Behandlung von Patienten qualitativ hochwertiger und effizienter zu machen ist es für die Mediziner notwendig, jederzeit und überall an die benötigten Informationen zu gelangen. Diese Mobilität soll durch eine auf einem mobilen Endgerät lauffähige Software erreicht werden, welche es den Mediziner*innen ermöglicht, die benötigte Daten abzufragen und ihre Erkenntnisse zu vermerken ohne dafür zuerst eine fest installierte Station aufsuchen zu müssen.

4.3.2 Produkt Funktion

Die Software ermöglicht es Mediziner*innen, anstehende Aufgaben abzufragen und deren Durchführung zu bestätigen. Ausserdem können Patientendaten von bestimmten Patienten gesucht und abgefragt werden.

4.3.3 Benutzer Charakteristik

Die Software richtet sich an medizinisches Fachpersonal, welches bereits mit ähnlichen Systemen vertraut ist, allerdings befinden sich diese auf fest installierten Arbeitsstationen bzw. medizinischen Geräten.

4.3.4 Einschränkungen

Obwohl der *DICOM*-Standard einen sehr grossen Funktionsumfang anbietet, wird für dieses Projekt nur ein kleiner Teil davon verwendet, damit die grundlegendsten und wichtigsten Funktionen unterstützt werden.

4.3.5 Abhängigkeiten

Da die Software den *DICOM*-Standard verwendet, sind die Grenzen durch diesen Standard gegeben. Die Funktionalität der Software ist also vom *DICOM*-Standard abhängig.

4.4 Funktionale Anforderungen

4.4.1 Muss Anforderungen

- Anmelden an einem zentralen *DICOM*-Server
- Abfragen von Behandlungsaufträgen («planned procedure steps») für verschiedene Untersuchungsstationen («application entities»)
- Bestätigen von durchgeführten Behandlungsaufträgen («performed procedure steps»), welche lediglich textuelle Resultate liefern
- Integration in bestehende Lösung [Vet09]

4.4.2 Optionale Anforderungen

Die optionalen Anforderungen sind nach Priorität sortiert:

1. Suchen/Abfragen von Patienteninformationen
2. Abfragen von durchgeführten Untersuchungen/Behandlungen («performed procedure steps») ohne Anzeige von Bildern
3. Abfragen von durchgeführten Behandlungsaufträgen mit Anzeige von Bilddateien
4. Lokales Cachen der Daten, damit Erfassungsaufgaben auch ohne eine Serververbindung möglich sind
5. Login-/Autorisierungsprozedur auf Clientseite um Datenmissbrauch auf lokaler Basis zu verhindern (mit Option auf zentralisierte Rechteverwaltung)
6. Verschlüsselung von lokal gespeicherten Daten
7. Update Mechanismus und zentrale Konfigurationsverwaltung

Für die funktionalen Anforderungen gilt auch noch das Kapitel 4.9 auf Seite 41.

4.5 Nicht funktionale Anforderungen

Die nicht funktionalen Anforderungen sind wie folgt für die Konformität kategorisiert:

- Kategorie 1: Müssen 100% richtig/erfüllt sein.
- Kategorie 2: Müssen >95% richtig/erfüllt sein.
- Kategorie 3: Müssen >90% richtig/erfüllt sein.

Die unter Konformität angesprochenen Testfälle werden in «Proof of Functionality and Testing» ausgewiesen/dokumentiert.

4.5.1 Funktionalität

4.5.1.1 Richtigkeit

Es werden die Patientendaten geliefert, welche verlangt werden, bzw. die «Planned procedure steps» für die richtige «application entity» dargestellt. Es gibt einen Schutzmechanismus welcher verhindert, dass eine Verwechslung durch gleiche Patientennamen auftritt. Diese Zuordnungen sind ausnahmslos richtig. Es kann anhand von Patientennamen gesucht werden. Sollte ein Name doppelt vorkommen, muss die richtige Person anhand zusätzlicher Daten (z. B. Adresse, Telefonnummer) validiert werden.

4.5.1.2 Interoperabilität

Die Software kann auf dem Zielsystem fehlerfrei betrieben werden und mit dem vorkonfigurierten *DICOM*-Server kommunizieren. Das Programm ist auf andere windowsbasierte Plattformen übertragbar, sofern die Mindestanforderungen berücksichtigt wurden.

4.5.1.3 Sicherheit

Die Software verwendet keine Dateien und Verzeichnisse, welche nicht zwingend für den Betrieb notwendig sind. Es dürfen keine Sicherheitslücken durch die Installation und den Betrieb der Software entstehen (z. B. durch offene Netzwerkverbindungen). Die Daten sind als medizinische Information besonders schützenswerte (Personen-)Daten und sind deshalb lokal und in der Übertragung zu verschlüsseln (siehe auch 4.5.1.4).

4.5.1.4 Ordnungsmässigkeit

Medizinische (Personen-)Daten sind nach Schweizer Gesetz besonders schützenswerte Personendaten. Es müssen deshalb Schutz- und Zugriffsmechanismen vorhanden sein, die dem schweizer Datenschutzgesetz entsprechen.

4.5.1.5 Konformität

Die Überprüfung der Funktionalitätspunkte obliegt dem Qualitätsmanagement. Folgende Prüfungstermine sind vorgesehen:

- Construction 2, Woche 2
- Construction 3, Woche 2

Die Konformität wird anhand einer Checkliste bestimmt und bildet einen Erfüllungsgrad.

- Richtigkeit: Kategorie 1
- Interoperabilität: Kategorie 2

- Sicherheit: Kategorie 1 (unter Einschränkungen bezüglich menschlichen Versagens)
- Ordnungsmässigkeit: Kategorie 1

4.5.2 Zuverlässigkeit

4.5.2.1 Fehlertoleranz

Die Software ist im Betrieb nicht vollständig von einer Netzwerkverbindung abhängig. Unterbrüche sind erlaubt und führen nicht zu Fehlverhalten und/oder Abstürzen. Dies bedeutet, dass der Stand eines «Procedure Steps» gespeichert (ausgeführt/nicht ausgeführt) und erneut oder verzögert gesendet werden kann. *DICOM*-Dokumente und Daten, welche nicht dem Standard entsprechen, führen nicht zu einem Fehlverhalten und/oder Abstürzen. Bei fehlerhaften Stellen wird entsprechend darauf hingewiesen, die restlichen Daten sind normal dargestellt.

4.5.2.2 Robustheit

Durch falsche Eingaben kann kein Fehlverhalten und/oder Abstürze provoziert werden. Das System fängt falsche Eingaben ab oder lässt diese nicht zu.

4.5.2.3 Wiederherstellbarkeit

Die Softwareeinstellungen sind lokal in einer Konfigurationsdatei abgespeichert und können schnell wieder hergestellt werden. Daten, die zum Zeitpunkt des Absturzes angezeigt wurden, können vom *DICOM*-Server neu geladen werden. Alle erledigten «Planned Procedure Steps» sind nicht verändert worden. Daten die durch einen Netzwerkunterbruch verloren gegangen sind, können durch den lokalen Cache wiederhergestellt werden. Nur vom Server bestätigte Daten gelten als empfangen.

4.5.2.4 Konformität

Die Überprüfung der Zuverlässigkeitspunkte obliegt dem Qualitätsmanagement. Folgende Prüfungstermine sind vorgesehen:

- Construction 3, Woche 1
- Construction 3, Woche 2

Die Konformität wird anhand einer Checkliste bestimmt und bildet einen Erfüllungsgrad, der bezüglich der Zuverlässigkeitspunkte einen mittleren Wert von 90 % erreichen muss.

- Fehlertoleranz: Kategorie 2
- Robustheit: Kategorie 2
- Wiederherstellbarkeit: Kategorie 2

4.5.3 Benutzbarkeit

4.5.3.1 Verständlichkeit

Die Benutzeroberfläche ist intuitiv und einfach zu verstehen bzw. zu bedienen. Es sind keine verwirrende Namen und Ausdrücke verwendet worden. Medizinische Fachausdrücke sind erlaubt/wünschenswert, da davon ausgegangen werden kann, dass sie dem Endbenutzer bekannt sind.

4.5.3.2 Erlernbarkeit

Die grundsätzliche Funktionalität, d. h. die Hauptbedienelemente und ihre Funktion zuzuordnen sowie eine grobe Navigationsübersicht ist dem Benutzer nach 4–7 Minuten klar. Dies beinhaltet das Abrufen der «Planned Procedure Steps» und das Suchen von Patientendaten. Weitere Funktionalitäten wie persönliche Einstellungen, Abrufen von Bildern etc. soll nach dem Studieren des Handbuchs möglich sein.

4.5.3.3 Bedienbarkeit

Die Steuerung der Software geschieht einerseits über die Touchscreen Funktionalität, andererseits über die zusätzlich vorhandene Tastatur. Suchanfragen für Patientendaten o. ä. geschehen über die Tastatureingabe, da dies komfortabler einzugeben ist als mittels einer Bildschirmtastatur. Die Navigation und das Abschliessen von «Planned Procedure Steps» wiederum geschehen über die Touchscreen Funktionalität.

4.5.3.4 Attraktivität

Die Attraktivität liegt darin, dass Arbeitsschritte effizienter und nahezu ortsunabhängig gehandhabt werden können. Die Arbeitsschritte können auf dem Gerät eingesehen und abgeschlossen werden, auch kann jederzeit eine Patienteninformation gesucht werden.

4.5.3.5 Konformität

Die Überprüfbarkeit der Benutzbarkeitspunkte obliegt den Usability Tests, beschrieben im Projektplan.

- Verständlichkeit: Kategorie 2
- Erlernbarkeit: Kategorie 3
- Bedienbarkeit: Kategorie 3
- Attraktivität: Kategorie 3

4.5.4 Effizienz

4.5.4.1 Zeitverhalten

Die Software reagiert innerhalb von 100 ms auf Eingaben und verarbeitet diese. Die maximale Ladezeit bei Datenanfragen an den zentralen *DICOM*-Server überschreiten sechs Sekunden nicht (sofern eine Verbindung existiert). Das Überspielen von zwischengespeicherten Daten dauert pro «Einheit» maximal 2 Sekunden.

4.5.4.2 Verbrauchs-/Ressourcenverhalten

Die Software muss auf der Zielplattform lauffähig sein (mobiles Gerät). Sie benötigt nicht mehr als 400 MB Speicherplatz und 128 MB RAM und lässt flüssiges Arbeiten zu.

4.5.4.3 Konformität

Die Überprüfbarkeit der Effizienzpunkte obliegt den System- und Performance Tests, beschrieben im Projektplan.

- Zeitverhalten: Kategorie 3
- Verbrauchs-/Ressourcenverhalten: Kategorie 2

4.5.5 Änderbarkeit

4.5.5.1 Stabilität

Änderungen von Implementierungen wirken sich nicht auf das Verhalten der Software aus. Komponenten treten isoliert auf, d. h. dass z. B. das Fehlen einer aktiven Netzwerkverbindung sich nicht auf die Stabilität der abhängigen Komponenten auswirkt.

4.5.5.2 Analysierbarkeit

Die Software besitzt einen Logging-Mechanismus, welcher es erlaubt, die Stelle eines Fehlers zu bestimmen und die möglichen Gründe einzuschränken. Der Aufwand zur Diagnose des Fehlers dauert bei schwerwiegenden Fehlern (verhindern den korrekten Einsatz der Software) maximal eine Stunde und erhöht sich linear in Abhängigkeit der Fehlerkategorie bis zu maximal einem Tag.

4.5.5.3 Modifizierbarkeit

Es gibt einen Updatemechanismus, der regelmässig überprüft, ob eine neue Version vorhanden ist. Es kann ein Update von der Serverseite auf das Gerät aufgespielt werden um z. B. Fehler zu beheben («Patch»).

4.5.5.4 Testbarkeit

Es werden die gleichen System-, Performance- und Usability Tests durchgeführt, wie im Projektplan beschrieben. Für neue Fähigkeiten der Software müssen die Testprotokolle/-listen entsprechend ergänzt werden. Dies benötigt maximal zwei Stunden.

4.5.5.5 Konformität

Die Überprüfbarkeit der Änderbarkeit wird durch die System- und Performance Tests gewährleistet. Der Update Mechanismus wird während der Entwicklung getestet und in die Usability - Tests integriert.

- Stabilität: Kategorie 2
- Analysierbarkeit: Kategorie 2
- Modifizierbarkeit: Kategorie 3
- Testbarkeit: Kategorie 3

4.5.6 Übertragbarkeit

4.5.6.1 Anpassbarkeit

Die Software kann ohne Anpassung auf Plattformen mit Java *ME*¹ betrieben werden, sofern diese die Hardware - Mindestanforderungen erfüllen.

4.5.6.2 Installierbarkeit

Der Installationsaufwand beträgt ohne individuelle Konfiguration (Benutzereinstellungen, welche die Bedienbarkeit und das Aussehen betreffen) maximal 15 Minuten. Die Installation geschieht mittels einer Anleitung und eines Installationspaketes.

4.5.6.3 Konformität

Die Übertragbarkeit wird durch die System- und Usability Tests überprüft.

- Anpassbarkeit: Kategorie 2
- Installierbarkeit: Kategorie 3

¹ *Micro Edition*

4.6 Schnittstellen

4.6.1 Benutzerschnittstelle

Die Benutzerschnittstelle ist eine grafische Oberfläche, welche an eigene Bedürfnisse angepasst werden kann (konkrete Beschreibungen sind für später vorbehalten). Sie wird mittels einer Tastatur oder eines Touchscreens angesteuert.

4.6.2 Hardwareschnittstelle

Die Hardwareschnittstelle ist durch die Benutzung von Java theoretisch transparent, da die Zielplattform jedoch ein mobiles Gerät ist, sind hier minimale Hardwareanforderungen zu beachten (siehe 4.5.4.2 auf Seite 37).

4.6.3 Softwareschnittstelle

Es wird die Testsoftware von *DVTk* benutzt und auch die entsprechenden Schnittstellen erarbeitet in [Vet09]. Die Zielplattform benötigt eine *JavaME* lauffähige Umgebung.

4.6.4 Datenbankschnittstelle

Es wird eine objekt- bzw. dateibasierte Serialisierung von Daten auf das Dateisystem benutzt (verschlüsselt).

4.6.5 Kommunikationsschnittstelle

Es wird über eine Punkt-zu-Punkt Verbindung zum Server gearbeitet. Der genaue Ablauf steht noch offen, entweder über die *java.net* und *java.nio* Netzwerkschnittstellen (direkt), oder aber über eine Bibliothek, welche dies schon integriert.

4.7 Lizenzanforderungen

Folgende Lizenzen werden benutzt/benötigt:

- *BSD*¹ [Tho01]
- Creative Common License [Cre09]

¹ *Berkeley Software Distribution*

4.8 Verwendete Standards

Folgende Standards werden verwendet:

- Java Coding Styleguide (mit Anpassungen)
- *ISO*¹/*IEC*² 9126
- *RUP* Software Engineering Process

¹ *Internationale Organisation für Normung* (von gr. : «isos»; zu dt. «gleich»)

² *Internationale elektrotechnische Kommission* (engl. *International Electrotechnical Commission*)

4.9 Use Cases

4.9.1 Diagramm

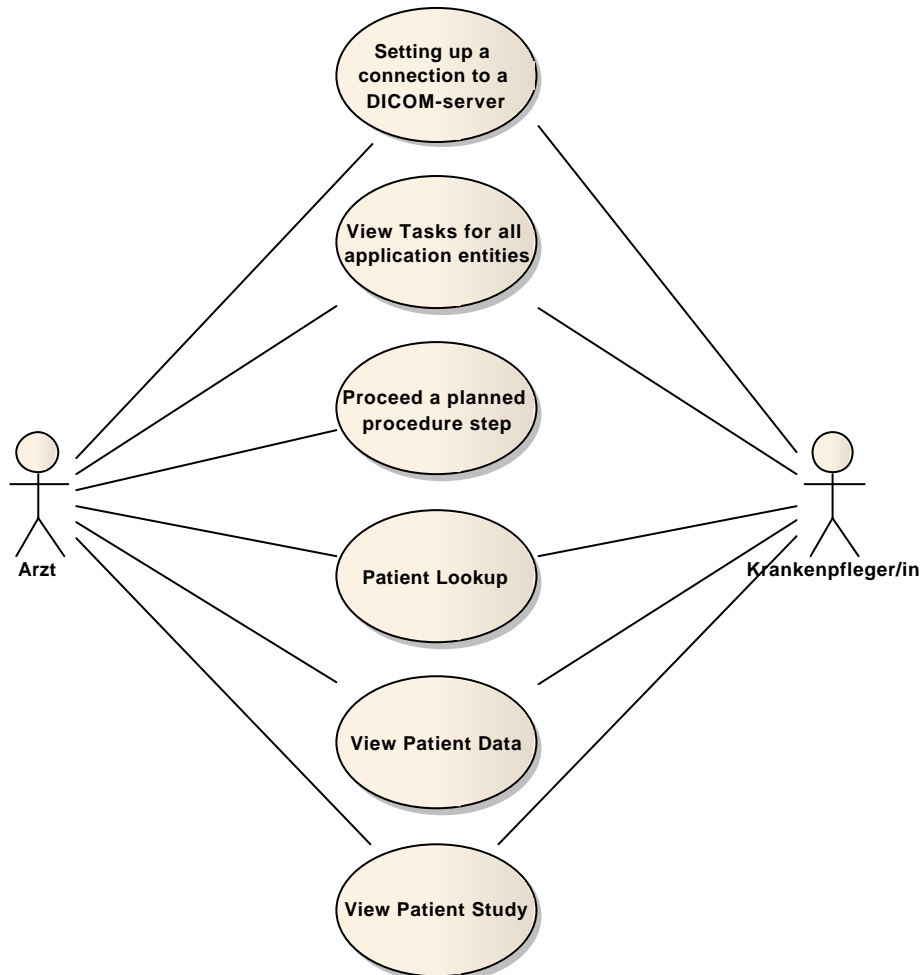


Abbildung (4.1) Use Case Diagramm

4.9.2 Brief

4.9.2.1 UC 1: Setting up a connection to a DICOM-server

Beschreibung: Ein Benutzer startet die Software und gibt die Verbindungsdaten zu einem *DICOM*-Server an, sofern noch keine konfiguriert sind.

4.9.2.2 UC 2: View Task for all «application entities»

Beschreibung: Ein Benutzer fragt eine Liste mit Behandlungsaufträgen ab.

4.9.2.3 UC 3: Proceed a Task

Beschreibung: Ein Benutzer wählt einen *Task* aus, trägt seine Resultate ein und schliesst den *Task* ab.

4.9.2.4 UC 4: Search for patient

Beschreibung: Ein Benutzer sucht nach den Patientendaten mit Hilfe des Patientennamens.

4.9.2.5 UC 5: View Patient Data

Beschreibung: Ein Benutzer lässt sich die Informationen eines Patienten anzeigen, z.B. Alter, Allergien, Krankheitsgeschichte usw.

4.9.2.6 UC 6: View Patient Study

Beschreibung: Ein Benutzer lässt sich die *Tasks* einer Behandlung anzeigen.

4.9.3 Aktoren

Arzt: Kann alle möglichen Aktionen durchführen, da er auch für Diagnosen das nötige Fachwissen hat.

Krankenpfleger/in: Kann nur Aktionen durchführen, die nicht so viel Fachwissen benötigen. Keine Diagnosen, dafür einfache Behandlungen

Administrator: Verwaltet den *DICOM*-Server und kann auf der Client-Seite keine Aktionen durchführen.

4.9.4 Stakeholders

4.9.4.1 Sicht Kunde

Die Sicht des Kunden bezieht sich auf den Auftraggeber des Projekts. Darunter werden einerseits – in diesem Rahmen – das Projektteam und die Betreuungsperson verstanden. Sie haben folgende Interessen:

<i>Stakeholder</i>	<i>Interesse</i>
Projektteam (mguhl, ythrier)	<ul style="list-style-type: none"> ▪ Gutes bis sehr gutes Resultat erreichen ▪ Image/Vorzeigematerial für spätere Zwecke
Betreuer (A. Doering)	<ul style="list-style-type: none"> ▪ Wissensvermittlung an Studierende ▪ Qualitativ hochstehendes Endresultat sowie Zwischenschritte als Ziel vorgeben

Tabelle (4.1) Stakeholders aus Kundensicht

4.9.4.2 Sicht Projektteam

Die Sicht des Projektteams bezieht sich auf die Mitarbeiter des Projekts und ihr persönliches Interesse an diesem Projekt (Profit, Erfahrung, Wissen etc.):

<i>Stakeholder</i>	<i>Interesse</i>
Projektteam (mguhl, ythrier)	<ul style="list-style-type: none"> ▪ Softwareentwicklungsprozess anhand <i>RUP</i> ▪ Erfahrungen Teamarbeit und Software-Entwicklung ▪ Erfahrungen im Programmieren von mobilen Kleingeräten ▪ Möglichst gute Note

Tabelle (4.2) Stakeholders aus Projektteamsicht

4.9.4.3 Sicht Endbenutzer

Die Sicht des Endbenutzers bezieht sich auf Personen, die das Endresultat schlussendlich benutzen werden. Ausgehend von einer medizinischen Fachperson sind nachfolgend ihre Interessen aufgelistet:

<i>Stakeholder</i>	<i>Interesse</i>
Medizinische Fachperson	<ul style="list-style-type: none"> ▪ Jederzeit/Überall über die nächsten Arbeitsschritte informiert sein ▪ Effizienteres Arbeiten ▪ Diagnosen erstellen ohne festinstallierte Arbeitsstationen aufsuchen zu müssen

Tabelle (4.3) Stakeholders aus Endbenutzersicht

4.9.5 Fully Dressed

4.9.5.1 UC 1: Setting up a connection to a DICOM-server

PrimaryActor: *Arzt*

Stakeholders and Interests:

- Krankenpfleger/in

Preconditions: -

Postconditions:

- Die Verbindungsdaten wurden gespeichert

Standard Ablauf

[1] **System:** Zeigt Eingabemasken für den Host und den Port des Servers an

[2] **User:** Trägt einen Host und einen Port ein

[3] **User:** Bestätigt seine Eingaben

[4] **System:** Speichert die Eingaben

[5] **System:** Kehrt zur vorherigen Anzeige zurück

Extensions

4a: Der Benutzer möchte die Einstellungen nicht übernehmen

[1] **User:** Teilt dem System mit, dass er die Daten verwerfen möchte

[2] **System:** Verwirft die Eingaben des Benutzers, weiter mit Step 5

4b: Der Benutzer hat ungültige Eingaben gemacht

[1] **System:** Zeigt dem Benutzer an, dass die Eingaben ungültig sind, weiter mit Step 2

Frequency of Occurrence: Beim ersten Start der Software und bei Änderung der Serververbindungsdaten

4.9.5.2 UC 2: View Task for all «application entities»

PrimaryActor: Arzt

Stakeholders and Interests:

- Krankenpfleger/in

Preconditions:

- Es besteht eine Netzwerkverbindung

Postconditions:

- Die Liste mit den *Tasks* wird angezeigt

Standard Ablauf

- [1] **User:** Fordert die Liste mit den *Tasks* an
- [2] **System:** Fordert die Liste beim zentralen Server an
- [3] **System:** Empfängt die Liste vom zentralen Server
- [4] **System:** Zeigt die Liste mit den *Tasks* an

Extensions

*a: Die Verbindung zum Server wurde unterbrochen

- [1] **System:** Informiert den Benutzer über den Verbindungsverlust
- [2] **System:** Fragt den Benutzer, ob er es nochmals versuchen oder lieber abbrechen möchte
- [3] **User:** Gibt an, wie er weiter vorgehen möchte

Frequency of Occurrence: Maximal alle 5 Minuten

4.9.5.3 UC 3: Proceed a Task

PrimaryActor: Arzt

Stakeholders and Interests: -

Preconditions:

- Es besteht eine Netzwerkverbindung
- Es wird eine Liste mit den *Tasks* angezeigt

Postconditions:

- Der *Task* wurde beim Server zusammen mit den Resultaten als erledigt eingetragen

Standard Ablauf

- [1] **User:** Wählt einen *Task* aus der Liste aus
- [2] **User:** Bestätigt, dass er den gewählten Step abschliessen möchte
- [3] **System:** Wechselt in eine Anzeige, in welcher der Benutzer seine Resultate eintragen kann
- [4] **User:** Gibt seine Resultate zu dem Step ein
- [5] **User:** Bestätigt seine Angaben
- [6] **System:** Sendet die eingegeben Daten zum Server
- [7] **System:** Bestätigt dem Benutzer, dass die Daten erfolgreich zum Server übertragen wurden

Extensions

6a: Die Daten konnten nicht an den Server übermittelt werden

[1] **System:** Informiert den Benutzer über den Fehlschlag

[2] **System:** Fragt den Benutzer, ob er es nochmals versuchen oder lieber abbrechen möchte

[3] **User:** Gibt an, wie er weiter vorgehen möchte

Frequency of Occurrence: Für jeden Step, welcher nicht zu einem Gerät gehört, ein Mal

4.9.5.4 UC 4: Search for patient

PrimaryActor: Arzt

Stakeholders and Interests:

- Krankenpfleger/in

Preconditions:

- Es besteht eine Netzwerkverbindung
- Die Suchansicht wird angezeigt

Postconditions:

- Die Behandlungen, welche der Patient bereits durchlaufen hat – oder gerade durchläuft – werden angezeigt

Standard Ablauf

[1] **User:** Gibt den Namen des gesuchten Patienten ein

[2] **User:** Startet den Suchauftrag

[3] **System:** Leitet den Suchauftrag an den Server weiter

[4] **System:** Empfängt das Suchresultat vom Server

[5] **System:** Fordert die Patientendaten des gefundenen Patienten vom Server an

[6] **System:** Empfängt die Patientendaten vom System

[7] **System:** Zeigt den Patienten an

Extensions

4a: Der Benutzer gibt statt des Patientennamens die Patientennummer ein

[1] **User:** Gibt die Patientennummer des gesuchten Patienten ein, weiter mit Step 2

4b: Es wurde kein Patient gefunden, auf den die Suchkriterien zutrifft

[1] **System:** Informiert den Benutzer darüber, dass kein entsprechender Patient gefunden wurde, weiter mit Step 1

4c: Es wurden mehrere Patienten gefunden, auf die die Suchkriterien zutreffen

[1] **System:** Zeigt dem Benutzer eine Liste , mit den gefundenen Patienten an

[2] **User:** Wählt den gewünschten Patienten aus, weiter mit Step 5

*d: Die Verbindung zum Server wurde unterbrochen

[1] **System:** Informiert den Benutzer über den Verbindungsverlust

[2] **System:** Fragt den Benutzer, ob er es nochmals versuchen oder lieber abbrechen möchte

[3] **User:** Gibt an, wie er weiter vorgehen möchte

Frequency of Occurrence: Pro Patient und Behandlung ein Mal

4.9.5.5 UC 5: View Patient Data

PrimaryActor: Arzt

Stakeholders and Interests:

- Krankenpfleger/in

Preconditions:

- Es besteht eine Netzwerkverbindung
- Es wurde ein Patient ausgewählt, der angezeigt werden soll mittels UC 4

Postconditions:

- Die gewünschten Detailinformationen des Patienten werden angezeigt

Standard Ablauf

[1] **User:** Fordert eine Liste von Informationen zum Patienten an

[2] **System:** Zeigt dem Benutzer eine Liste von Patienteninformationen an

[3] **User:** Wählt die gewünschte Information aus

[4] **System:** Fordert die gewünschte Informationen vom Server an

[5] **System:** Empfängt die gewünschte Information vom Server

[6] **System:** Zeigt die Information an

Extensions

*a: Die Verbindung zum Server wurde unterbrochen

[1] **System:** Informiert den Benutzer über den Verbindungsverlust

[2] **System:** Fragt den Benutzer, ob er es nochmals versuchen oder lieber abbrechen möchte

[3] **User:** Gibt an, wie er weiter vorgehen möchte

Frequency of Occurrence: Ein paar Male pro Patient und Behandlung

4.9.5.6 UC 6: View Patient Study

PrimaryActor: Arzt

Stakeholders and Interests:

- Krankenpfleger/in

Preconditions:

- Es besteht eine Verbindung mit dem zentralen Server
- Es wurde ein Patient ausgewählt, der angezeigt werden soll mittels UC 4

Postconditions:

- Die *Tasks* der gewünschten Behandlung werden angezeigt

Standard Ablauf

[1] **User:** Fordert eine Liste der Behandlungen des Patienten an

[2] **System:** Zeigt dem Benutzer eine Liste der Behandlungen an

[3] **User:** Wählt eine Behandlung aus

[4] **System:** Fordert die *Tasks* der gewünschten Behandlung vom Server an

[5] **System:** Empfängt die *Tasks* vom Server

[6] **System:** Zeigt die *Tasks* an

Extensions

*a: Die Verbindung zum Server wurde unterbrochen

[1] **System:** Informiert den Benutzer über den Verbindungsverlust

[2] **System:** Fragt den Benutzer, ob er es nochmals versuchen oder lieber abbrechen möchte

[3] **User:** Gibt an, wie er weiter vorgehen möchte

Frequency of Occurrence: Pro Patient und Behandlung ein Mal

5 Domain Analyse

5.1 Dokument

5.1.1 Dokumentinformationen

Version: 1.0
Revision: Rev: 417
Status: Release
Erstellt am: 20.10.2009
Erstellt von: Mario Guhl

5.1.2 Dokumenthistory

Rev.	Datum	Wer	Änderung
0	20.10.2009	mguhl	Dokument aufgesetzt
105	21.10.2009	mguhl	Domain-Model und Kapitelüberschriften hinzugefügt
106	21.10.2009	ythrier	SSDs und Konzepte hinzugefügt
108	21.10.2009	mguhl	Dokument überarbeitet und Contracts hinzugefügt
143	29.10.2009	ythrier	SSDs Update / Review
143	31.10.2009	mguhl	Dokument neu strukturiert, Paper-Prototype hinzugefügt, neue/überarbeitet Sequenzdiagramme eingefügt, State Diagramm bei UC3 eingefügt, Domain-Model leicht überarbeitet gemäss Team-Meeting vom 29.10.2009
149	31.10.2009	ythrier	SSD for UC1 (Geändert von connection establish zu connection configuration)
150	31.10.2009	mguhl	Contracts überarbeitet, State Diagramm überarbeitet
153	01.11.2009	ythrier	Logische-/Technologieschichten eingefügt. SSDs Verbindungsaufbau (beinhaltet auch die Abbruchszenarien)
158	01.11.2009	ythrier	DICOM Einführung für ein besseres Verständnis der Domäne (notwendig da es nicht möglich ist, rein durch die Domain-Analyse den Standard zu verstehen)
159	01.11.2009	mguhl	Contracts für UC1 erstellt, GUI-Prototype überarbeitet
165	01.11.2009	ythrier	DICOM-Analyse mit Assoziierungsaufbau und Schichten.
206	28.11.2009	ythrier	Label für Referenzierung in SAD

5.2 Einführung

5.2.1 Zweck

Dieses Dokument beschreibt die Domain Architektur.

5.2.2 Gültigkeitsbereich

Dieses Dokument ist für das gesamte Projekt über die gesamte Bearbeitungszeit gültig.

5.2.3 Übersicht

Es wird zuerst eine kurze Einführung in *DICOM* gegeben, danach das Domainmodell erklärt und darauf folgt für jeden Use Case ein *SSD*¹ mit den dazugehörigen Contracts. Schlussendlich folgt noch die *GUI*²- und *DICOM*-Analyse.

5.3 *DICOM*-Einführung

Um nachfolgende Sektionen besser verstehen zu können, werden hier einige wichtigen Punkte bezüglich des *DICOM*-Standards aufgegriffen und erläutert.

5.3.1 Was ist *DICOM*?

DICOM ist ein offener Standard zum Austausch von Informationen in der Medizin. Diese Informationen können beispielsweise auch Röntgenbilder sein oder Patienteninformationen. Der Standard definiert hierbei auch, wie ein Verbindung aufgebaut werden muss. Dies beinhaltet unter anderem die Aushandlung eines «Transfer Syntaxes». Weiter sind Datenformate und Tags definiert, welche einzelne Identifikationen darstellen. Es gibt auch sogenannte Service-Klassen, welche einen Dienst beschreiben, d. h. den Funktionsumfang und die zu unterstützenden Punkte des Standards.

1 *System Sequenz Diagramm*

2 *Graphical User Interface*

5.3.2 Das Informationsmodell

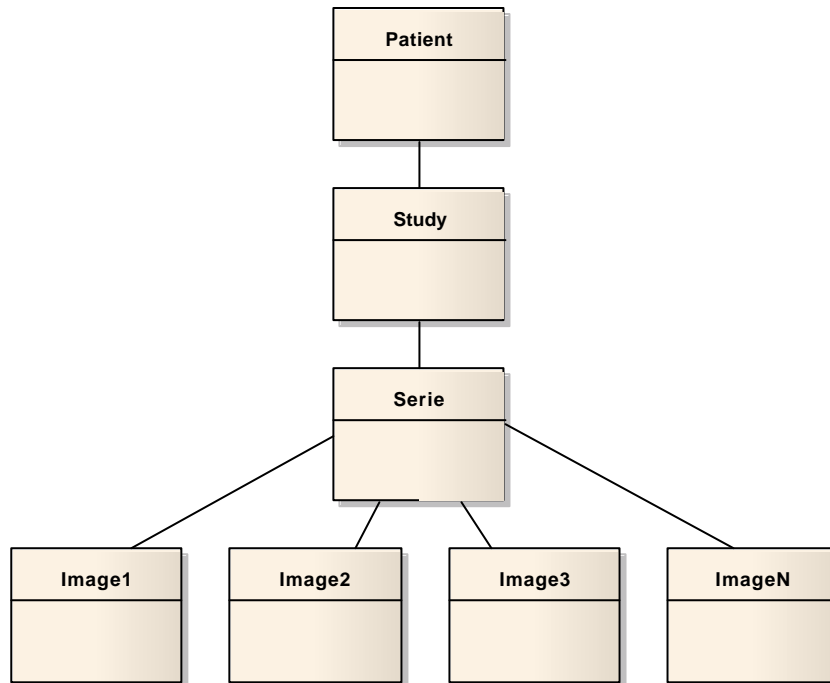


Abbildung (5.1) Diagramm Informationsmodell

Das Informationsmodell beschreibt die Darstellung der *DICOM* Daten. Ein Patient korreliert zu einem realen Patient. Eine Study kann als ein «Arztbesuch» angesehen werden. Ein Arztbesuch kann hierbei ein oder mehrere Ergebnisse liefern, beispielsweise Röntgenbilder. Diese werden in Series zusammengefasst.

5.3.3 Service-Klassen

Der Standard definiert Service-Klassen, welche bestimmte Funktionalitäten übernehmen. Zum Beispiel gibt es eine Service-Klasse, welche *DICOM* Daten auf einem Server abspeichert. Clientseitig spricht man hier von einem «Service-Class-User» (*SCU*¹), also der Service-Benutzer. Der Server wird hier als «Service-Class-Provider» (*SCP*²) bezeichnet (Dienstanbieter). Ein *SCU* und *SCP* des gleichen Services bezeichnet man als «Service-Object-Pair» (*SOP*³).

1 *Service Class User*

2 *Service Class Provider*

3 *Service Object Pair*

5.3.4 Composite und Normalized

Composite und Normalized sind Funktionsgruppen. Z. B. gibt es eine Methode C(omposite)-Find, welche für das Abfragen von Arbeitsschritten auf einem *DICOM*-Server benutzt wird.

5.3.5 Verbindungsaufbau

Der Verbindungsaufbau geschieht zwischen einem *SCU* und *SCP* und startet mit einer Assoziierung. In der Assoziierung wird dem *SCP* ein «Abstract Syntax» und ein oder mehrere «Transfer Syntax'» angegeben (dies nennt man «Presentation Context»). Der Abstract Syntax beschreibt hierbei, was man tun möchte, als eine Identifikation für die Art des Services. Transfer Syntax ist die Datendarstellung für die Übertragung. Es gibt hier beispielsweise Little- und Big-Endian.

5.3.6 Tags

Tags sind Identifikatoren, welche z. B. einen Nachrichtentyp beschreiben (z. B. Assoziierungsanfrage) oder Datenfelder (z. B. gibt es einen Tag welcher indiziert, dass nun ein Abstract Syntax kommt).

5.4 Domainmodell

5.4.1 Strukturdiagramm

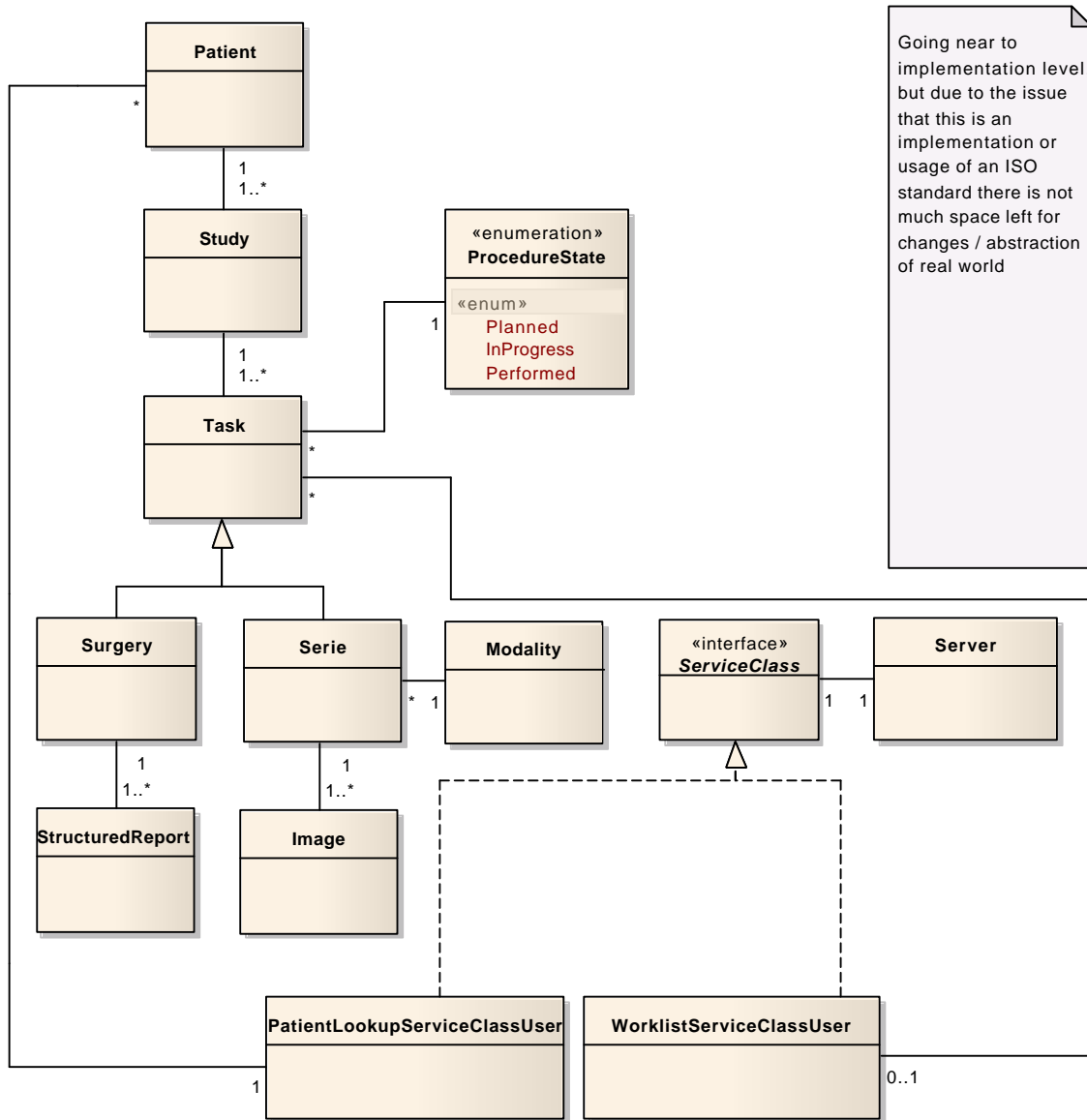


Abbildung (5.2) Diagramm Domainmodell

5.4.2 Konzeptbeschreibung

5.4.2.1 Patient

Ein Patient repräsentiert eine Person, d. h. ein realer Patient. Er hat verschiedene Eigenschaften wie Name, Alter, Geschlecht, Allergien etc. Dies entspricht dem Patient im *DICOM*-Informationsmodell.

5.4.2.2 Study

Eine Study ist eine «Sitzung», welche mit einem Patient durchgeführt wurde. Dies kann eine Untersuchung mit Diagnose oder auch das Erstellen von Bildern an Modalitäten sein. Eine Study hat eine Identifikation und ist einem Patienten zugeordnet. Dies entspricht der Study im *DICOM*-Informationsmodell.

5.4.2.3 Task

Ein *Task* wird hier als Oberbegriff für einen Arzt-/Spitalbesuch verwendet und kann eine Diagnose sein oder auch erstellte Bilder an einer Modalität.

5.4.2.4 Surgery

Eine Surgery ist ein spezifischer *Task*, dessen Daten (z. B. eine Diagnose) in einem Report dargestellt werden.

5.4.2.5 StructuredReport

Ein StructuredReport ist eine Datendarstellung von z. B. einer Diagnose oder Informationen über den Patienten. Dies entspricht dem StructuredReport von *DICOM*.

5.4.2.6 Serie

Die Serie ist ebenfalls ein spezifischer *Task*, dessen Daten sich in Bildern (bzw. allgemeinen Daten) von Modalitäten darstellen. Entspricht der Serie im *DICOM*-Informationsmodell.

5.4.2.7 Image

Ein Bild, erstellt von einer Modalität (o. a.).

5.4.2.8 ProcedureState

Ein *Task* hat einen Status. Aus Sicht des Patientensuchdienstes ist dies «Performed», aus Sicht des Worklistdienstes wird dies zu Beginn «Planned» sein und über «InProgress» zu «Performed» traversieren.

5.4.2.9 Modality

Repräsentiert eine Modalität. Korreliert zur Modality im *DICOM*-Informationsmodell.

5.4.2.10 ServiceClass

Die ServiceClass stellt eine Schnittstelle für die angebotenen Services (Worklist und Patientensuche) dar.

5.4.2.11 PatientLookupServiceClassUser

Ist der Patientensuchdienst über den Patienten gesucht werden können.

5.4.2.12 WorklistServiceClassUser

Ist der Worklistdienst, über den die «Planned Procedure Steps» für eine Modalität oder einen Patienten geholt werden können.

5.4.2.13 NetworkConnection

Die NetworkConnection ist die Schnittstelle zum *DICOM*-Server, auf dem die Worklist und die Patientendaten bereit liegen.

5.5 Systemoperationen/*System Sequenz Diagramme*

5.5.1 Contract/*SSD* für UC1: Setting up a connection to a *DICOM*-server

5.5.1.1 Sequenzdiagramm

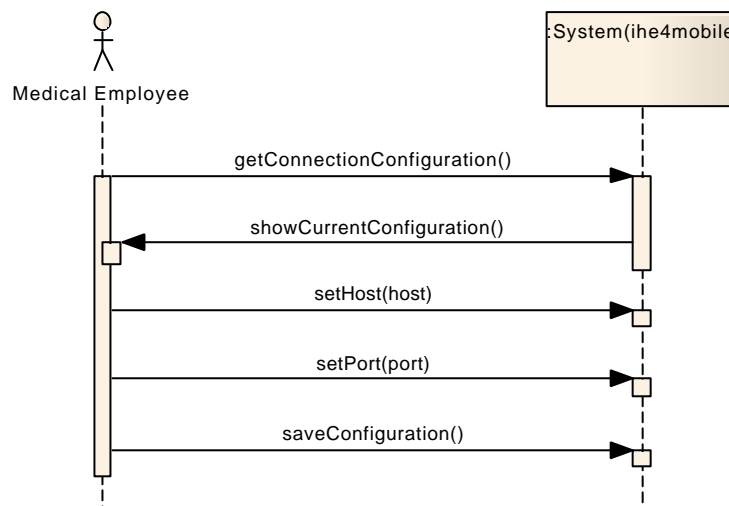


Abbildung (5.3) *SSD* für UC1: Setting up a connection to a *DICOM*-server

5.5.1.2 Contract: GetConnectionStringConfiguration

Operation: getConnectionStringConfiguration()

Preconditions: -

Postconditions:

- Die aktuellen Verbindungseinstellungen wurden zurückgegeben

5.5.1.3 Contract: SetHost

Operation: setHost(host)

Preconditions:

- Es wurde ein gültiger Hostname angegeben

Postconditions:

- Der Hostname wurde übernommen

5.5.1.4 Contract: SetPort

Operation:	setPort(port)
Preconditions:	<ul style="list-style-type: none">▪ Es wurde ein gültiger Port angegeben
Postconditions:	<ul style="list-style-type: none">▪ Der Port wurde übernommen

5.5.1.5 Contract: SaveConfiguration

Operation:	saveConfiguration()
Preconditions:	<ul style="list-style-type: none">▪ Es wurden Änderungen am Host/Port vorgenommen
Postconditions:	<ul style="list-style-type: none">▪ Die neue Konfiguration wurde gespeichert

5.5.2 Contract/SSD für UC2: View Task for all «application entities»

5.5.2.1 Sequenzdiagramm

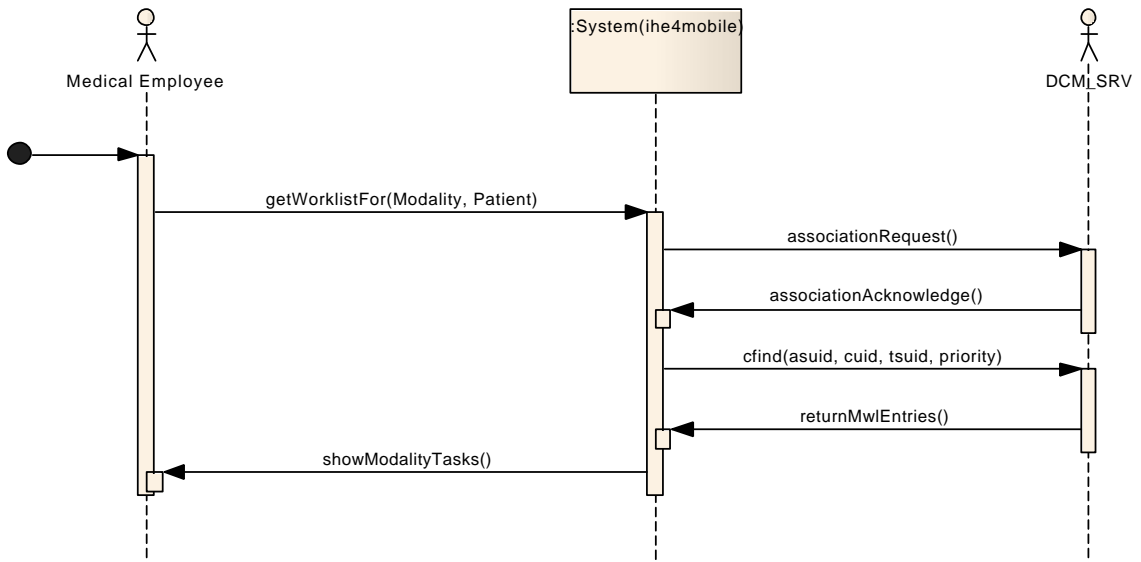


Abbildung (5.4) SSD für UC2: View Task for all «application entities»

5.5.2.2 Contract: GetWorklistFor

Operation:	getWorklistFor(modality, patient)
Preconditions:	<ul style="list-style-type: none"> ▪ Es besteht eine Netzwerkverbindung ▪ Eine Modalität und/oder ein Patient wurde angegeben
Postconditions:	<ul style="list-style-type: none"> ▪ Eine Liste der gefundenen <i>Tasks</i> wurde zurückgegeben

5.5.3 Contract/SSD für UC3: Proceed a Task

5.5.3.1 Sequenzdiagramm

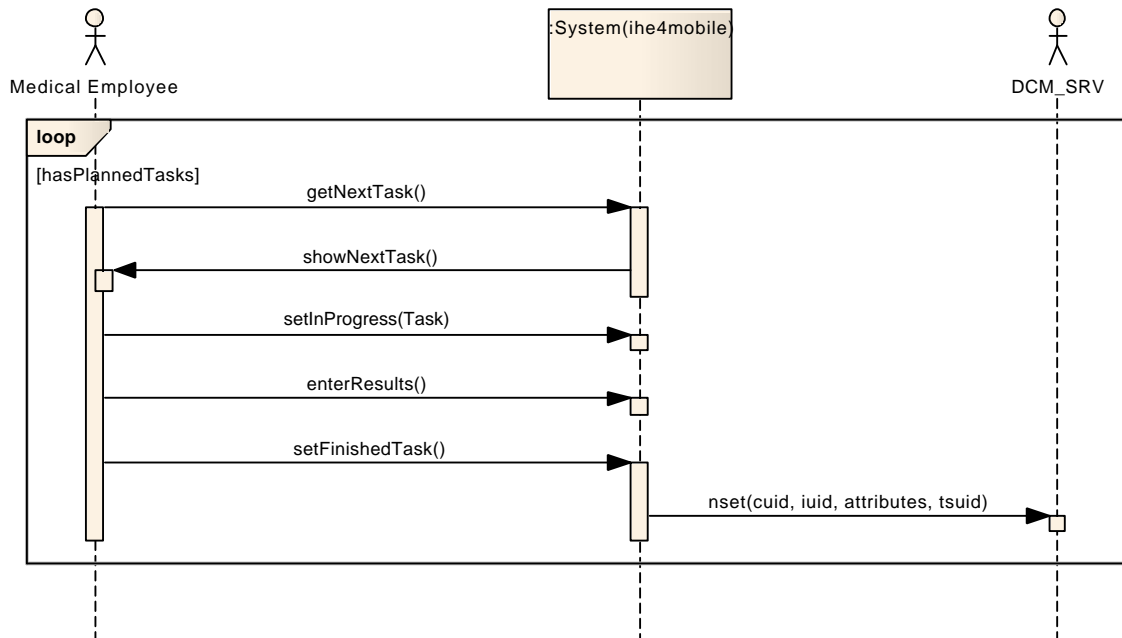


Abbildung (5.5) SSD für UC3: Proceed a Task

5.5.3.2 Statediagramm

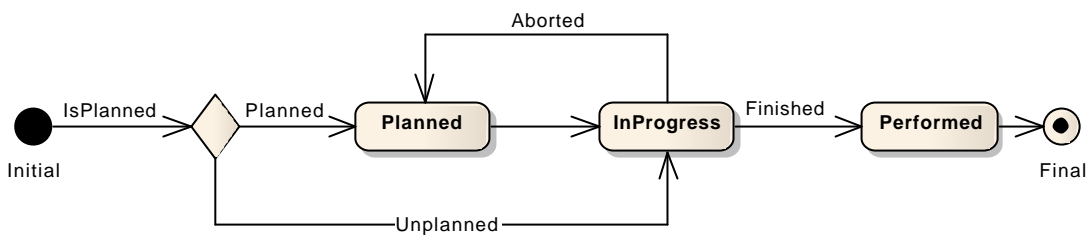


Abbildung (5.6) Zustände eines Tasks

5.5.3.3 Contract: GetNextTask

Operation:	getNextTask()
Preconditions:	<ul style="list-style-type: none"> ▪ Es wurde vorher eine Liste von Tasks abgerufen
Postconditions:	<ul style="list-style-type: none"> ▪ Der nächste Task wurde zurückgegeben

5.5.3.4 Contract: SetInProgress

Operation:	setInProgress()
Preconditions:	<ul style="list-style-type: none">▪ Es besteht eine Netzwerkverbindung▪ Es wurde ein existierender <i>Task</i> angegeben
Postconditions:	<ul style="list-style-type: none">▪ Der Status des <i>Tasks</i> wurde auf «InProgress» gesetzt

5.5.3.5 Contract: EnterResults

Operation:	enterResults()
Preconditions:	<ul style="list-style-type: none">▪ Es wurde vorher ein <i>Task</i> auf «InProgress» gesetzt
Postconditions:	<ul style="list-style-type: none">▪ Die Resultate wurden dem <i>Task</i> hinzugefügt

5.5.3.6 Contract: SetFinishedTask

Operation:	setFinishedTask()
Preconditions:	<ul style="list-style-type: none">▪ Es besteht eine Netzwerkverbindung▪ Es wurde vorher ein <i>Task</i> auf «InProgress» gesetzt
Postconditions:	<ul style="list-style-type: none">▪ Der Status des <i>Tasks</i> wurde auf «Performed» gesetzt

5.5.4 Contract/SSD für UC4: Search for patient

5.5.4.1 Sequenzdiagramm

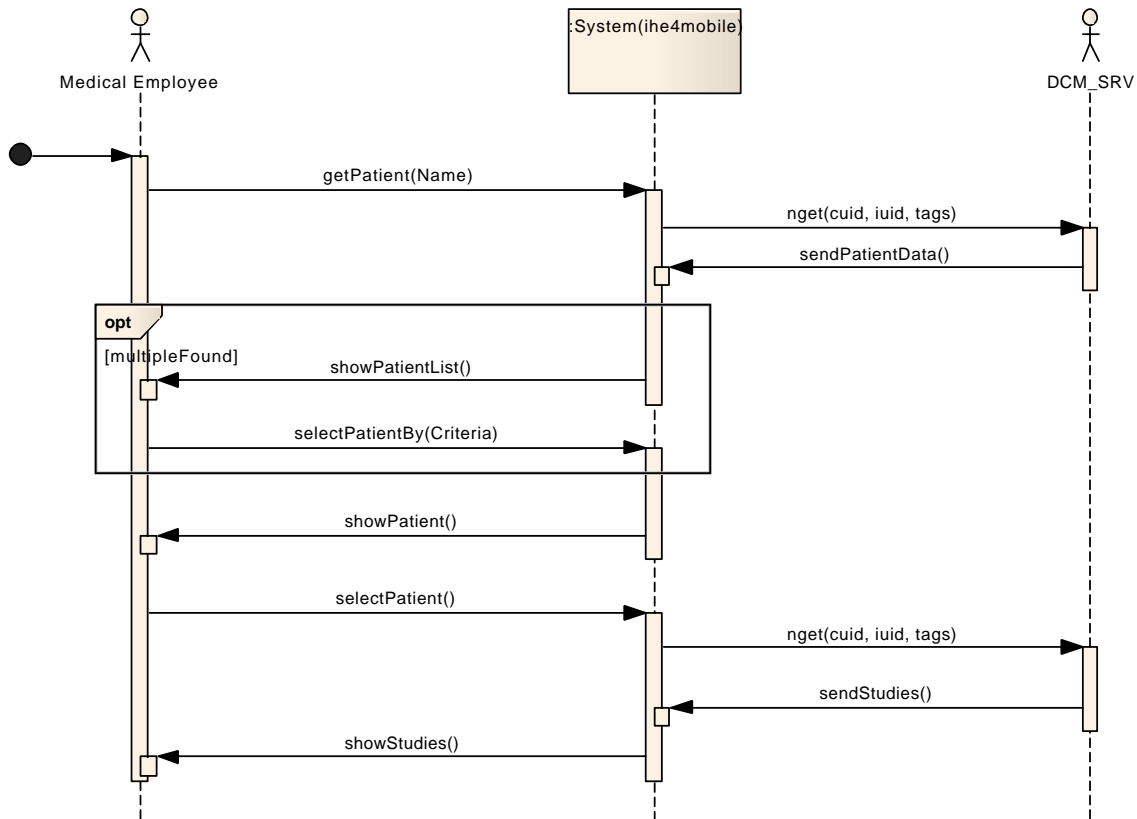


Abbildung (5.7) SSD für UC4: Search Patient

5.5.4.2 Contract: GetPatient

Operation:	getPatient(Name)
Preconditions:	<ul style="list-style-type: none"> ▪ Es besteht eine Netzwerkverbindung
Postconditions:	<ul style="list-style-type: none"> ▪ Der Patient mit dem gesuchten Namen wurde zurückgegeben ▪ Wenn mehrere Patienten mit dem Namen gefunden wurden, wurde eine Liste der Patienten mit dem gesuchten Namen zurückgeben

5.5.4.3 Contract: SelectPatientBy

Operation: selectPatientBy(Criteria)

Preconditions:

- Es besteht eine Netzwerkverbindung
- Es wurde vorher eine Suchanfrage nach einem Patientennamen durchgeführt

Postconditions:

- Der Patient, auf den die Kriterien zutreffen, wurde zurückgegeben

5.5.4.4 Contract: SelectPatient

Operation: selectPatient()

Preconditions:

- Es besteht eine Netzwerkverbindung
- Es wurde vorher ein Patient ausgewählt

Postconditions:

- Die Studies des Patienten wurden zurückgegeben

5.5.5 Contract/SSD für UC5: View Patient Data

5.5.5.1 Sequenzdiagramm

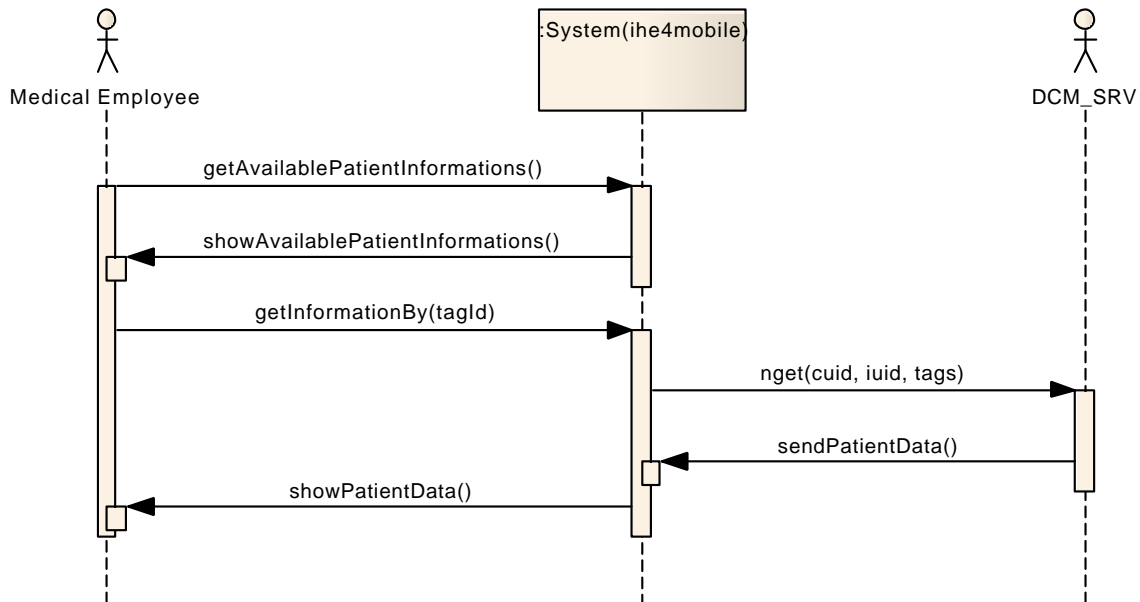


Abbildung (5.8) SSD für UC5: View Patient Data

5.5.5.2 Contract: GetAvailablePatientInformations

Operation:	getAvailablePatientInformations()
Preconditions:	<ul style="list-style-type: none"> ▪ Es besteht eine Netzwerkverbindung ▪ Es wurde vorher ein Patient ausgewählt
Postconditions:	<ul style="list-style-type: none"> ▪ Eine Liste mit den vorhandenen Informationen zum Patienten wurde zurückgegeben

5.5.5.3 Contract: GetInformationBy

Operation:	getInformationBy(tagId)
Preconditions:	<ul style="list-style-type: none"> ▪ Es besteht eine Netzwerkverbindung ▪ Es wurde vorher ein Patient ausgewählt ▪ Es wurde eine gültige tagId angegeben
Postconditions:	<ul style="list-style-type: none"> ▪ Die Informationen mit der entsprechenden tagId wurde zurückgegeben

5.6 Logische-/Technologieschichten

5.6.1 Technologie-Layers

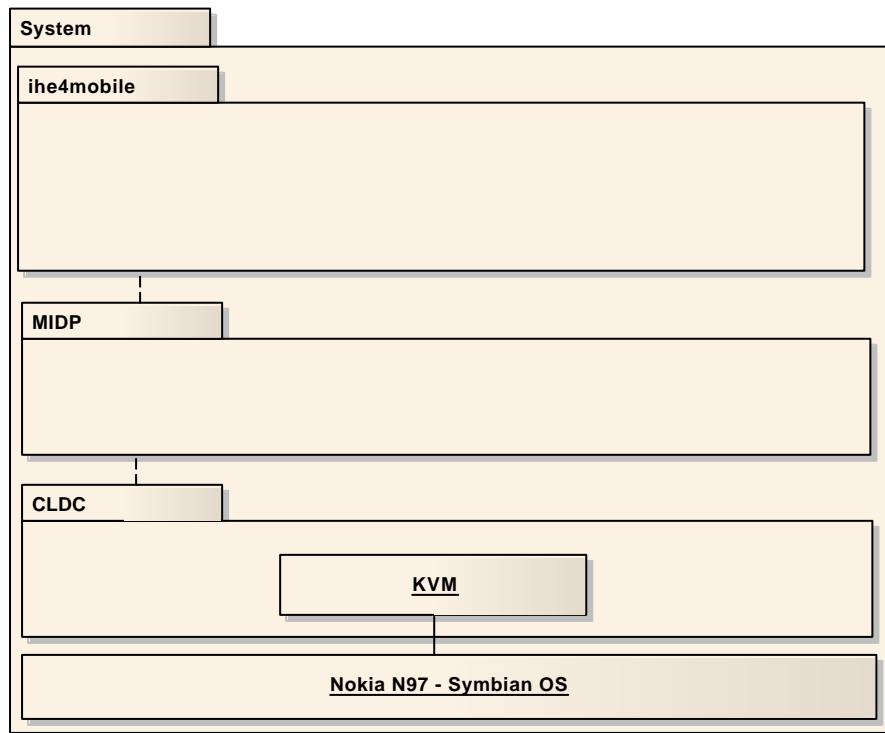


Abbildung (5.9) Diagramm Technologieschichten

5.6.1.1 ihe4mobile

Das zu entwickelnde System. Enthält die einzelnen Benutzerschnittstellen, Verbindungsaufbau sowie Persistence und auch die Geschäftslogik.

5.6.1.2 MIDP

«Mobile Information Device Profile». Oberschicht, welche in der Java Mobile Edition enthalten ist. Erlaubt es Applikationsentwicklern, Programme für netzwerkfähige Geräte zu schreiben.

5.6.1.3 CLDC

«Connected Limited Device Configuration». Basisset der Programmierschnittstellen für Geräte mit Batterie (Akku) und Speichereinschränkungen (insb. mobile Geräte).

5.6.1.4 KVM

«K-Virtual Machine». Kompakte Java VM für Plattformen mit Ressourceneinschränkungen (Speicher, Batterie etc.).

5.7 GUI-Analyse

5.7.1 Paper-Prototype

5.7.1.1 Hauptmenü

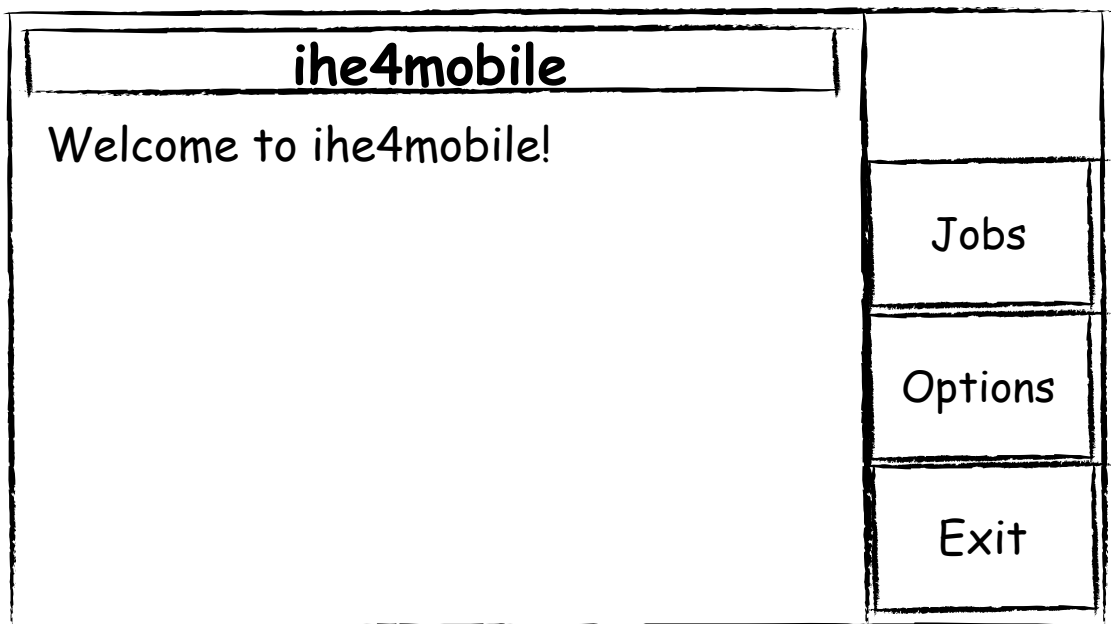


Abbildung (5.10) Hauptmenü

Wird direkt nach dem Starten der Anwendung angezeigt. Von hier aus können verschiedene Aufgaben (Jobs) erledigt werden oder Einstellungen (Options) vorgenommen werden.

5.7.1.2 Taskliste

Tasklist	
Task 1	
Task 2	
Task 3	
Task 4	Navigation
Task 5	
Task 6	Exit

Abbildung (5.11) Taskliste

Zeigt eine Liste von *Tasks* an. Durch Antippen eines *Tasks* werden die Details des *Tasks* angezeigt (siehe 5.7.1.3 auf der nächsten Seite). Zudem kommt man von hier aus in die Programmoptionen.

5.7.1.3 *Task* Details

Task1 Details	
Patient: Jon Doe	
Modality: X-Ray	Complete Task
Description: X-ray the leg	
Report: This is a dummy text. This...	Navigation
	Exit

Abbildung (5.12) *Task* Details

Hier werden detaillierte Informationen zu dem *Task* angezeigt, z. B. der Name des Patienten, für den der *Task* ist oder die Modalität mit welcher der *Task* durchgeführt werden soll, sofern es eine gibt. Ausserdem wird der zum *Task* gehörende Report angezeigt, welcher durch Antippen auch bearbeitet werden kann (siehe 5.7.1.4 auf der nächsten Seite). Ausserdem kann der *Task* als «Proceeded» gekennzeichnet werden.

5.7.1.4 Report bearbeiten

Edit Report	
This is a dummy text. This is a dummy text. This is a dummy text. This is a dummy text. This is a dummy text. This is a dummy text. This is a dummy text. This is a dummy text.	
	Save
	Cancel
	Exit

Abbildung (5.13) Report bearbeiten

In diesem Screen kann ein Report verfasst und überarbeitet werden, am Ende kann er entweder gespeichert oder verworfen werden.

5.7.1.5 Optionen

Options	
Host:	<input type="text" value="DICOM_SERVER"/>
Port:	<input type="text" value="10000"/>
Cyphering-Key:	<input type="text" value="*****"/>
Key-Repetition:	<input type="text" value="*****"/>
	<input type="button" value="Save"/>
	<input type="button" value="Cancel"/>
	<input type="button" value="Exit"/>

Abbildung (5.14) Optionen

Wird angezeigt, wenn man vom Hauptmenü (siehe 5.7.1.1 auf Seite 67) die Optionen auswählt. Hier können der Host und Port des Servers eingestellt werden und ein Passwort, mit dem lokal gespeicherte Daten verschlüsselt werden.

5.7.2 User Environment Model

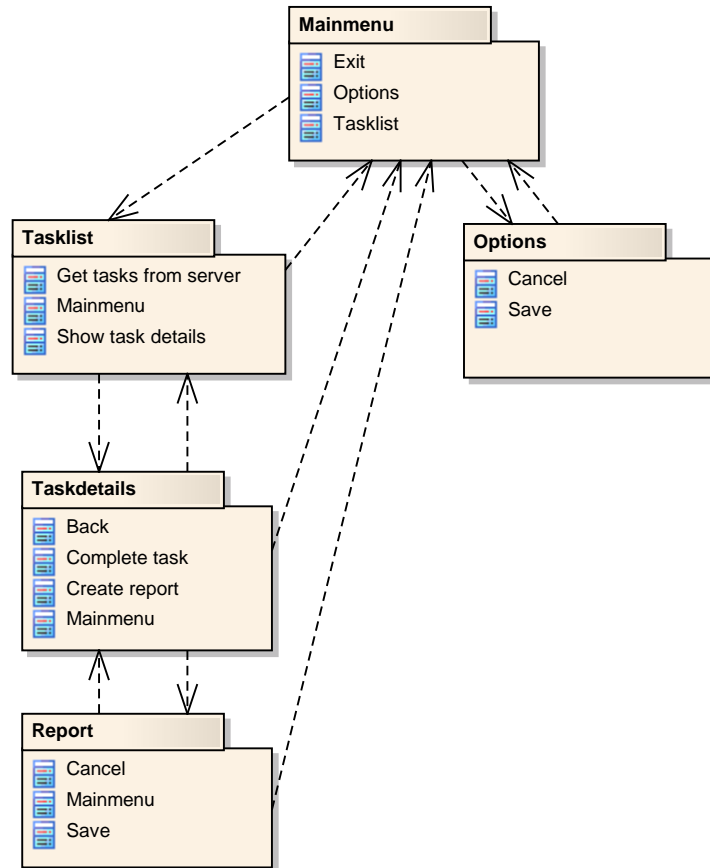


Abbildung (5.15) User Environment Model

5.8 DICOM-Analyse

5.8.1 Übersicht

Da die ursprünglich geplante Library nicht auf Java Mobile Edition einsetzbar ist, müssen die benötigten Funktionalitäten in das System portiert werden. Nachfolgend wird dies aus Analysesicht betrachtet.

5.8.2 Operationen

5.8.2.1 Association

Die Assoziierung muss zu Beginn erfolgen und beinhaltet das Aushandeln von Übertragungs-Syntaxen und das Festlegen der Operation (Composite-Find oder Normalized-Set) mittels der abstrakten Syntax. Zuerst muss ein Association Request vom *SCU* an den *SCP* geschickt werden. Dieser antwortet normalerweise mit einem Association Acknowledge, welcher die unterstützten Übertragungs-Syntaxen enthält. Sollte der *SCP* keine Verbindungen mehr annehmen können, schickt er ein Association Reject. Sollte der Server keinen Übertragungs-Syntax des Clients unterstützen, können keine Daten geschickt werden, deshalb muss der Client die Assoziierung mit einem Association Abort beenden. Bei einer erfolgreichen Assoziierung kann danach ein Composite Find oder ein Normalized Set ausgeführt werden.

Association Ablauf

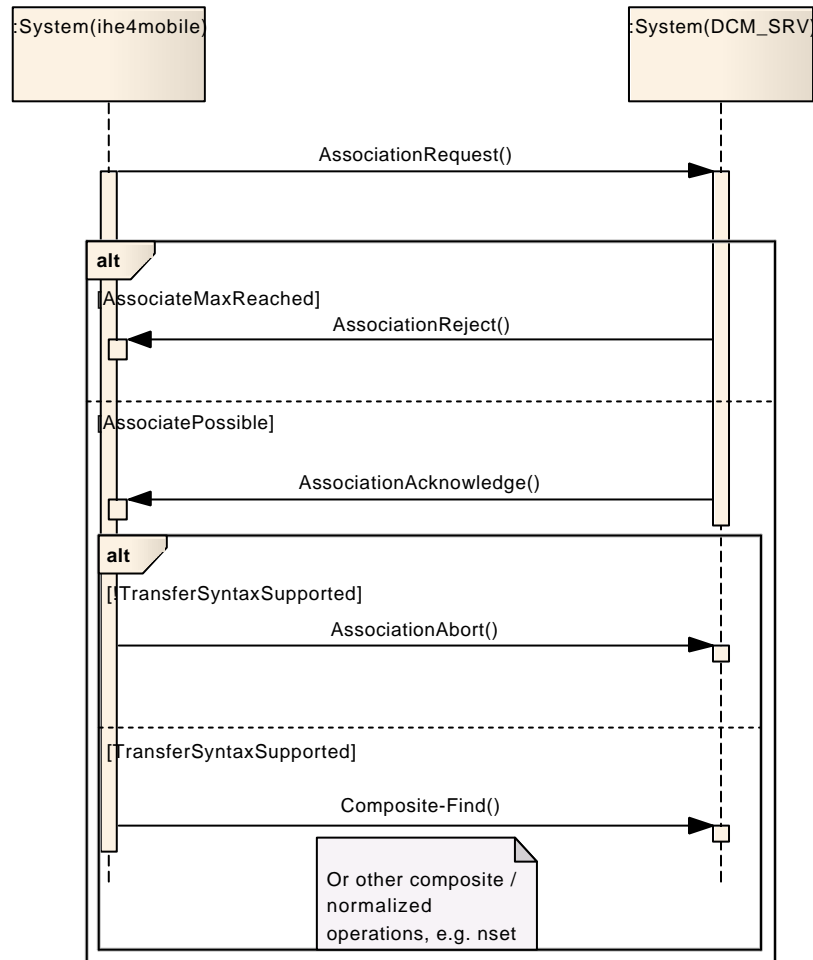


Abbildung (5.16) Diagramm Association Ablauf

Association Request/Acknowledge Nachrichtenaufbau

Beispiel einer Nachricht (die Acknowledge Nachricht hat denselben Aufbau, anhand der Transfer Syntaxen erkennt man welche unterstützt werden).

Field	Size (Byte)
Message tag	1
Padding	1
Request length	4
Padding	2

weiter auf der nächsten Seite

<i>Field</i>	<i>Size (Byte)</i>
Called entity	16
Calling entity	16
Reserved	32
Context ag	1
Padding	2
Context length	2
Context tag	individual
Presentation context tag	1
Padding	1
Presentation context length	2
Reserved	1
Padding	3
Abstract syntax tag	1
Padding	1
Abstract syntax length	2
Transfer syntax 1 tag	1
Padding	1
Transfer syntax 1 length	2
Transfer syntax	individual
User identification tag	1
Padding	1
User identification length	2
Max Pdu size tag	1
Padding	1
Max Pdu size length	2
Max Pdu size	4
Class identification tag	1
Padding	1
Class identification length	2
Class identification	individual
Implementation identification tag	1
Padding	1
Implementation identification length	2
Implementation identification	individual

Tabelle (5.1) Association Request/Acknowledge Nachrichtenaufbau

5.8.2.2 Composite Find

Composite Find ist die Operation, welche benötigt wird, um Arbeitsschritte zu einem Patienten für eine Modalität abzurufen. Die Operation muss dabei eine Modalitätsidentifikation und eine Patientenidentifikation an den *SCP* schicken, welcher dann die verfügbaren Arbeitsschritte zurückschickt.

5.8.2.3 Normalized Set

Normalized Set ist die Operation, welche benötigt wird, um Arbeitsschritte zu einem Patienten für eine Modalität als erledigt zu markieren. Die Operation nimmt dabei Bezug auf einen vorherig mit Composite Find abgerufenen Arbeitsschritt.

5.8.3 Portierbarkeit

Um unabhängig von Java Mobile Edition die zu implementierenden Funktionen auch anderweitig später benutzen zu können, wird es eine generelle «Connection»-Schnittstelle geben, über die jede mögliche Art von Verbindung implementiert werden kann (z. B. für Java *ME* eine *StreamConnection*, für Java 1.6 *PC* Plattformen ein normaler *Socket*).

5.8.4 Schichten

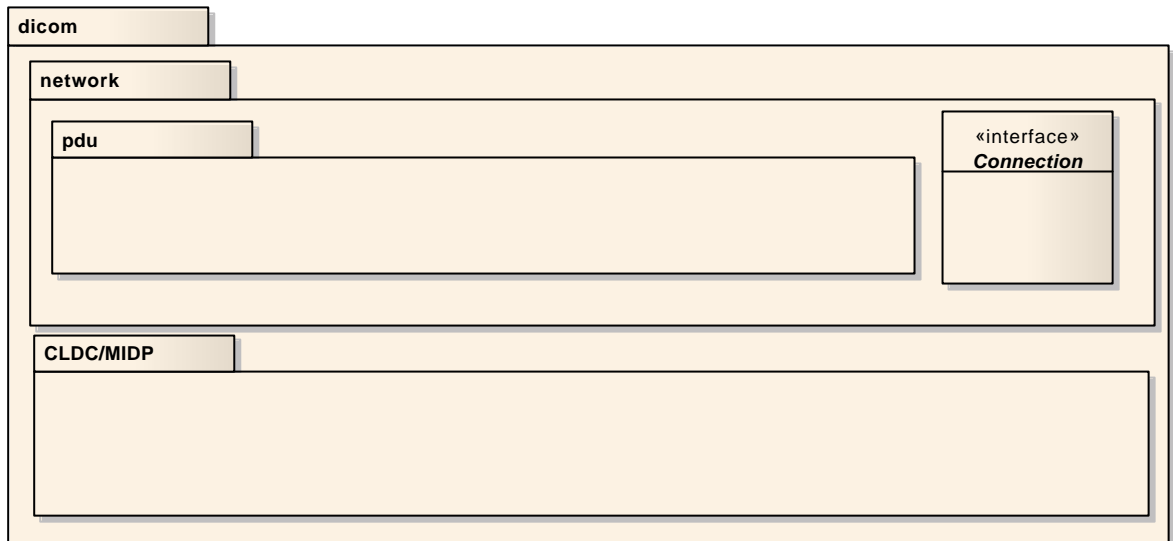


Abbildung (5.17) Diagramm *DICOM* Netzwerk Schichten

6 Software Architektur Dokument

6.1 Dokument

6.1.1 Dokumentinformationen

Version: 1.0
Revision: Rev: 416
Status: Release
Erstellt am: 20.09.2009
Erstellt von: Mario Guhl

6.1.2 Dokumenthistory

Rev.	Datum	Wer	Änderung
0	20.09.2009	mguhl	Dokument aufgesetzt
132	27.09.2009	mguhl	Beschreibung der Layer hinzugefügt
143	01.11.2009	mguhl	<i>GUI</i> -Package beschrieben
161	01.11.2009	mguhl	Business-Package beschrieben (noch nicht fertig)
194	08.11.2009	mguhl	Kapitel Client/(Server) überarbeitet, Package <i>GUI</i> überarbeitet, Package Business fertiggestellt, Package Persistence angefangen
206	28.11.2009	ythrier	Network Struktur und Erklärungen
298	29.11.2009	ythrier	VR und PDU Package, Parser begonnen
302	30.11.2009	ythrier/mguhl	Tabellengrößen geändert, damit sie mit Textbreite übereinstimmen
388	15.12.2009	ythrier/mguhl	Klassendiagramme hinzugefügt
390	15.12.2009	ythrier/mguhl	Deployment

6.2 Einführung

6.2.1 Zweck

Dieses Dokument dient der Beschreibung der Architektur dieser Software.

6.2.2 Gültigkeitsbereich

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

6.2.3 Übersicht

In diesem Dokument wird die logische Architektur der Software beschrieben. Es werden sowohl der strukturelle Aufbau als auch die Schichten detailliert beschrieben. Ausserdem enthält es *UML*¹ sowie Sequenzdiagramme, welche die Übersichtlichkeit erhöhen.

6.3 Architektonische Darstellung

Hier wird eine Übersicht über alle Schichten gegeben und die Konzepte beschrieben, welche sich über alle Schichten hinwegziehen.

6.3.1 SSD über alle Schichten

6.3.2 3-Layer Architektur

Die Software wird als 3-Layer Architektur umgesetzt. Die drei Layer bestehen aus *GUI*, Business-Layer und Persistence-Layer. Der *GUI*-Layer greift nur auf den Business-Layer zu, um die Daten aus dem Persistence-Layer darzustellen. Der Business-Layer wiederum greift auf den Persistence-Layer zu und bereitet die Daten dann auf. Der Persistence-Layer ist in zwei Teile aufgeteilt, ein Teil ist für die lokale Datenorganisation zuständig (Storage) und der andere Teil für den Netzwerkzugriff auf die Daten.

6.3.3 Client/(Server)

Die Software wird mit dem Client/Server-Modell umgesetzt. Wobei der Server-Teil bereits besteht und die Software demnach nur noch den Client-Teil implementiert. Auf dem Server sind alle Daten gespeichert und können vom Client abgerufen werden. Dabei kann der Client bei der Abfrage Kriterien definieren, nach denen die Daten ausgewählt werden. Dadurch übernimmt der (im Vergleich zum mobilen Endgerät) leistungsstarke Server die Filterung und die Daten werden auch wesentlich schneller übertragen als wenn alles übertragen würde und der Client die Daten filtern müsste. Der Client speichert keine Daten persistent ab, ausser die Daten, welche gerade bearbeitet werden, damit sie im Falle eines Ausfalls wiederhergestellt werden können. Die Aufgaben des Clients bestehen also im Wesentlichen darin, die Daten darzustellen. Ausserdem ermöglicht es der Client noch, Reports zu erstellen und *Tasks* abzuschliessen.

1 *Unified Modeling Language*

6.3.3.1 Datenfilterung

Damit möglichst wenig Daten übertragen werden müssen, werden Detailinformationen zu Datensätzen erst dann abgerufen, wenn sie wirklich benötigt werden. Wird z. B. der Datensatz eines Patienten abgerufen, wird zuerst nach dem Namen/Patientennummer gefiltert, dann wird eine Liste der zutreffenden Namen übermittelt und erst bei Auswahl eines Namens wird der gesamte Datensatz übermittelt. Bei den *Tasks* wird ebenfalls nur eine Liste mit der Bezeichnung des *Tasks* angezeigt und erst bei Auswahl eines bestimmten *Tasks* wird die Beschreibung usw. übermittelt.

6.3.3.2 GUI-Synchronisation

Damit das *GUI* bei länger dauernden Vorgängen, wie z. B. Serveranfragen nicht blockiert, bietet der Business-Layer entsprechende Methoden an um Methoden asynchron auszuführen und bietet Listener an, um das *GUI* über den Abschluss einer Operation zu informieren. Dies ermöglicht es auch, dem Benutzer bei bestimmten Operationen die Möglichkeit zu geben die Operation abubrechen, falls sie zu lange dauert.

6.4 Architektonische Ziele und Einschränkungen

6.4.1 Ziele

- Zwischen zwei Layern darf es nur Abhängigkeiten in eine Richtung geben.
- Es darf nur Abhängigkeiten zwischen «benachbarten» Layern geben, der *GUI*-Layer darf also z. B. nicht auf den Persistence-Layer zugreifen.
- Die Software soll stabil laufen und fähig sein, noch nicht an den Server übertragene Daten im Falle eines Softwarefehlers oder Ausfalls z. B. aufgrund eines leeren Akkus wiederherzustellen.
- Das *GUI* sollte sehr einfach gehalten sein, damit es auf dem kleinen Display eines mobilen Endgerätes gut bedienbar ist.

6.4.2 Einschränkungen

- Die Software geht von einem korrekt funktionierenden Server aus, welche keine fehlerhaft codierten Daten liefert. Die Software darf zwar nicht abstürzen, jedoch ist ein korrektes Weiterarbeiten ohne Neustart der Software nicht garantiert.
- Die Software ist nur bei vorhandener Verbindung mit dem Server lauffähig. Optional kann aber auch Offline-Funktionalität eingebaut werden (siehe Kapitel 4.4.2 auf Seite 33).

6.5 Logische Architektur

6.5.1 Übersicht

Um die Software in verschiedene Aufgabengebiete einzuteilen wurde sie in verschiedene Layer aufgeteilt. Dies ermöglicht ein unabhängiges Entwickeln der verschiedenen Teile.

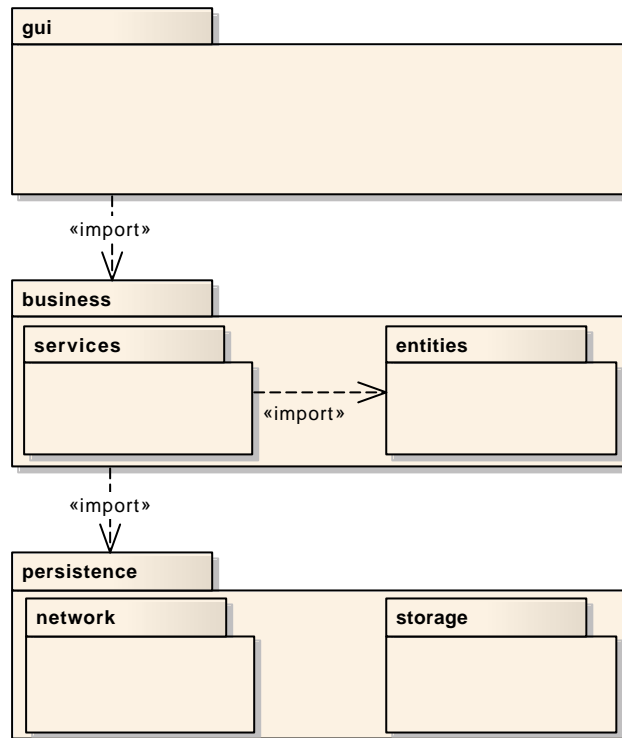


Abbildung (6.1) Layeraufteilung

6.5.1.1 SSD über alle Layer

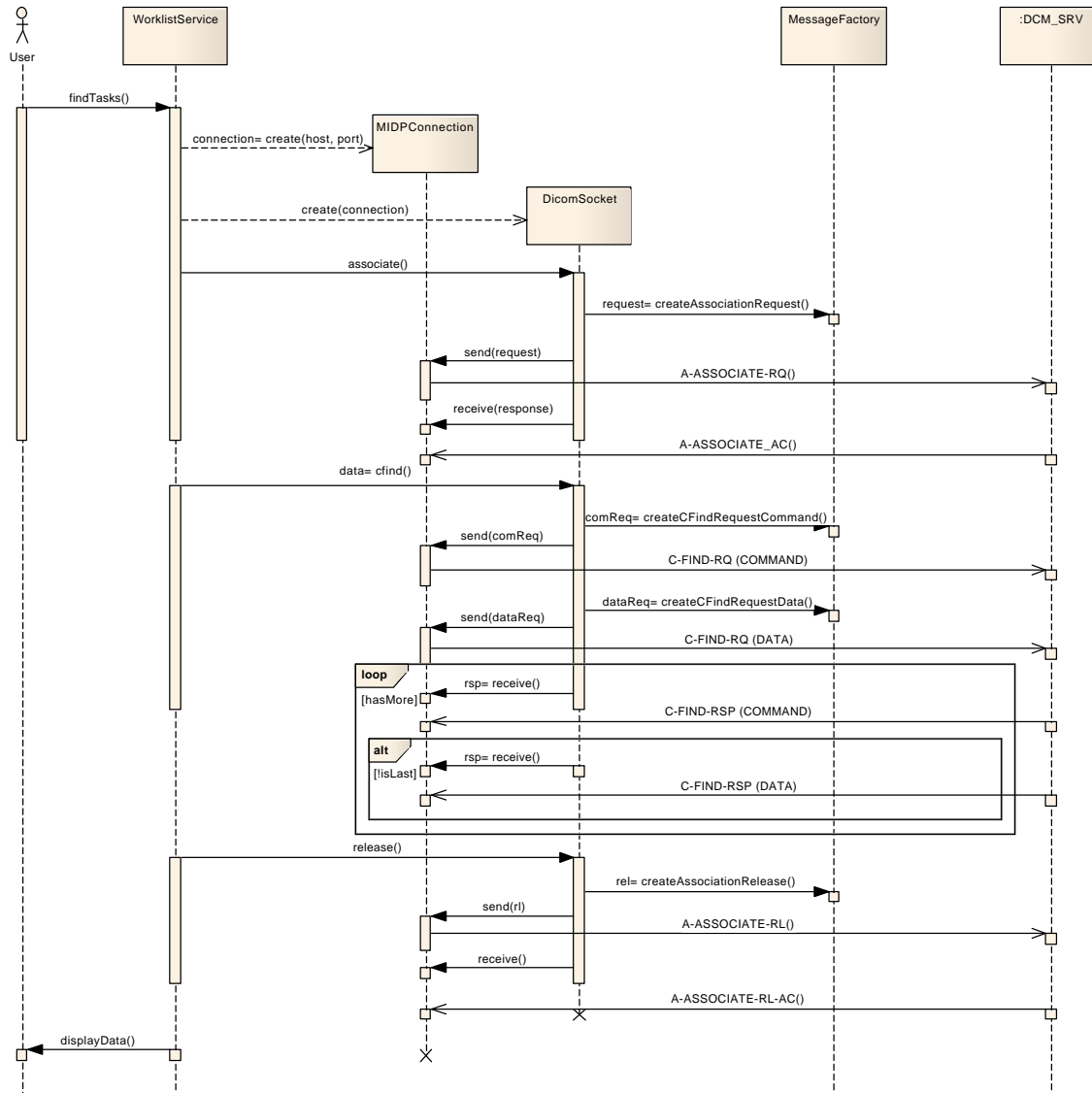


Abbildung (6.2) SSD über alle Layer

6.5.1.2 GUI

Das wichtigste am *GUI* ist, dass es sehr kompakt und übersichtlich aufgebaut ist, da auf dem Display eines mobilen Endgerätes die Möglichkeiten sehr eingeschränkt sind. Dies führt auch dazu, dass dieses Package einen eher kleinen Teil der Software ausmacht.

Ausserdem muss das *GUI* so aufgebaut sein, dass es durch Operationen, welche länger dauern nicht blockieren kann und es jederzeit möglich ist, die Operation abzubrechen. Dies wird dadurch bewerkstelligt, dass nur asynchrone Operationen benutzt werden, wenn es sich um länger dauernde Operationen handelt, wie z. B. Netzwerkoperationen.

6.5.1.3 Business

In diesem Package wird die eigentliche Logik der Software umgesetzt. Auch wird der grösste Teil des Domain-Modells im Business-Package implementiert. Einerseits enthält es Funktionen um verschiedene Abfragen auf dem *DICOM*-Server auszuführen, andererseits bildet es verschiedene Entitäten des *DICOM*- und *HL7*-Modells ab. Zudem werden alle Methoden in diesem Package, welche auch eine längere Verarbeitungszeit haben können, wie z. B. Netzwerkoperationen, so implementiert, dass sie auch asynchron aufgerufen werden können. Somit kann das *GUI* einfach nicht-blockierend umgesetzt werden, wenn es die entsprechenden Methoden verwendet.

Services

In diesem Package befinden sich verschiedene Services, mit denen Abfragen auf dem Server durchgeführt werden können. Es stellt Methoden zur Verfügung um Informationen über Patienten oder geplante *Tasks* abzurufen. Ausserdem stellt es Funktionalitäten zur Verfügung um Daten, welche gerade bearbeitet werden, persistent abzuspeichern, um sie im Falle eines Ausfalls wiederherstellen zu können.

Entities

Dieses Package stellt verschiedene Klassen zur Verfügung um die verschiedenen Entitäten des *DICOM*- bzw. *HL7*-Modells abzubilden. Dazu gehören z. B. *Tasks*, Patienten, Modalitäten usw. Deshalb bietet diese Package auch nicht viel Funktionalität, sondern dient mehr der Abbildung der Daten nach dem Domain-Model (siehe Kapitel 5.2 auf Seite 55).

6.5.1.4 Storage

Dieses Package bietet verschiedene Möglichkeiten an, Daten zu persistieren. Einerseits über das Netzwerk, andererseits auf dem lokalen, nichtflüchtigen Speicher. Dazu bietet das Package einheitliche Schnittstellen für Speicher und Netzwerk an. So wäre es z. B. möglich, die Daten vor dem Speichern noch zu verschlüsseln, sofern dieses Feature zeitlich noch eingebaut werden kann.

Network

Dieses Package kapselt den relativ komplizierten Verbindungsaufbau zu einem *DICOM*-Server in einfach zu verwendende Klassen. Die Schnittstellen werden so aufgebaut, dass es möglich ist, die Daten durch Zwischenschalten von Modulen, z. B. für Verschlüsselung oder Komprimierung, vor dem Übertragen beliebig zu verändern.

Persistence

Das Persistence-Package bietet Funktionen an, mit denen beliebige Daten auf lokalem Speicher abgelegt werden. Durch einheitliche Schnittstellen soll das Speichern der Daten ohne Kenntnisse des benutzten Speichers möglich sein und es soll auch hier möglich sein, die Daten vorher mit verschiedenen Modulen zu verarbeiten.

6.5.2 Package *GUI*

6.5.2.1 Beschreibung des Package

Das Package ist die Schnittstelle zwischen dem Benutzer und der Problem-Domain der Applikation. Es visualisiert die Daten der Problem-Domain und leitet Befehle des Benutzers an die Business-Logic weiter. Ausserdem werden die Klassen der einzelnen Packages in diesem Package initialisiert und miteinander verknüpft.

6.5.2.2 Diagramm

Da das *GUI*-Package aufgrund der Zielhardware sehr klein gehalten werden muss, spiegelt das Klassendiagramm lediglich die verschiedenen Screens wider, welche im Paper-Prototype (siehe Kapitel 5.7.1 auf Seite 67) definiert wurden.

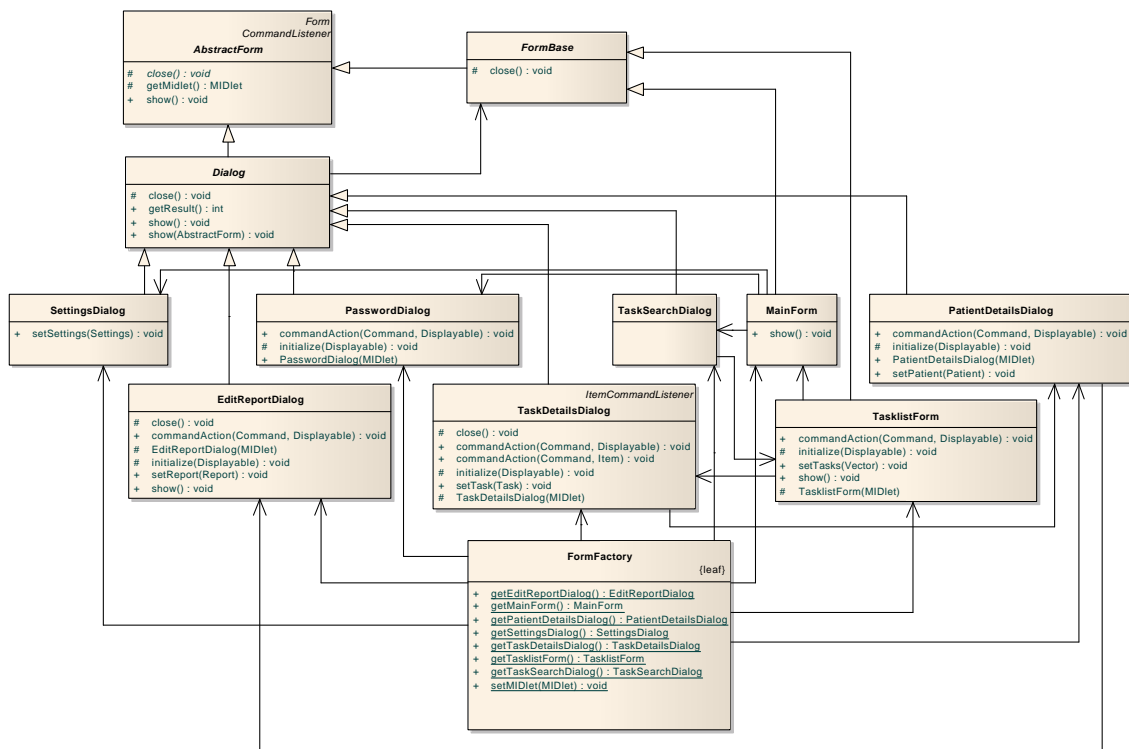


Abbildung (6.3) Klassendiagramm Package GUI

6.5.2.3 Schnittstellen

Das GUI-Package stellt keine Schnittstellen zur Verfügung, da es den obersten Layer darstellt und deshalb nur auf die Schnittstellen der anderen Packages zugreift.

6.5.2.4 Klassen

AbstractForm Stellt die wichtigsten Funktionalitäten zur Verfügung, welche bei einer Form benötigt werden. Sie implementiert das CommandListener-Interface, welches die Verarbeitung von Befehlen (z. B. Exit, Save, Cancel usw.) ermöglicht, speichert das *MIDlet*, welches die Form besitzt und führt die abstrakte Methode `exit` ein. Das *MIDlet* wird benötigt um die Applikation zu beenden. Je nachdem um was für eine Form es sich handelt, soll die Applikation aber beim Schliessen der Form nicht beendet werden weshalb, die EXIT-Methode nur abstrakt ist.

FormBase Stellt eine Form dar, welche sich ähnlich verhält wie eine Form des Java GUIs für Computer. Sie implementiert die EXIT-Methode so, dass die Applikation beendet wird, wenn sie aufgerufen wird. Ausserdem stellt sie auch gleich einen Exit-Befehl auf

dem *GUI* dar, über den der Befehl aufgerufen wird. Zudem wird die `show`-Methode zur Verfügung gestellt, welche dazu führt, dass die Form angezeigt wird.

Dialog Ist genau gesehen ebenfalls eine Form wie aus dem Klassendiagramm (siehe Abbildung 6.3 auf der vorherigen Seite) ersichtlich. Verhält sich aber ähnlich wie ein Dialog des Java *GUIs* für Computer. Beim Aufruf der `exit`-Methode wird nämlich nicht die Applikation beendet, sondern die Form angezeigt, welche den Dialog aufgerufen hat. Deshalb muss in der `show`-Methode auch die aufrufende Form mitgegeben werden.

MainForm Stellt die Hauptanzeige des *GUIs* dar. Wenn das Passwort noch nicht eingegeben wurde, wird zuerst der `PasswordDialog` aufgerufen und dieses mit Hilfe der `Settings`-Klasse (siehe 6.5.3.5 auf Seite 88) geprüft. Sorgt dafür, dass die Servereinstellungen gesetzt werden, falls sie noch nicht gesetzt wurden. Dazu wird nach dem Initialisieren der Klasse und somit nach dem Starten der Applikation über die `Business-Logic` abgefragt, ob die Servereinstellungen schon gesetzt sind und das `Optionsmenü` angezeigt wird, falls sie noch nicht gesetzt sind. Die Klasse ermöglicht es ausserdem zu den Einstellungen zu navigieren und eine *Taskliste* anzuzeigen. Für die *Taskliste* wird aber nicht direkt die `TasklistForm` angezeigt, sondern zuerst ein Dialog, in dem eingestellt werden kann, welche *Tasks* angezeigt werden sollen.

TaskSearchDialog Ermöglicht das Einstellen des Patienten und/oder der Modalität zu dem/der *Tasks* angezeigt werden soll. Ausserdem kann die Suche dann eingeleitet werden, worauf die gefundenen *Tasks* auf der `TasklistForm` angezeigt werden oder der Suchvorgang kann abgebrochen werden, worauf wieder die aufrufende Form, in diesem Fall also die `MainForm`, angezeigt wird.

TasklistForm Greift auf die `WorklistService`-Klasse (siehe 6.5.3.5 auf Seite 88) des `Business-Layers` zu, um eine Liste der *Tasks* zu erhalten. Leitet die Information zu einem Task an den `TaskDetailsFrame` weiter, damit dieser die Detailinformationen anzeigen kann. Ermöglicht ausserdem die Navigation zu anderen Aufgaben (optionale Features) und zurück zum Hauptmenü.

TaskDetailsDialog Zeigt die Detailinformationen zu einem Task an. Ermöglicht es einen Task als erledigt zu kennzeichnen und benutzt dazu die `WorklistService`-Klasse. Sendet ebenfalls über die `WorklistService`-Klasse einen Report an den Server nachdem er mittels Aufruf des `ReportFrames` erstellt/bearbeitet wurde. Ermöglicht ausserdem die Navigation zu anderen Aufgaben (z. B. *Tasklist*) oder zurück zur Hauptanzeige.

PatientDetailsDialog Zeigt Detaillierte Informationen zu einem Patienten an. Ermöglicht die Navigation zurück zum `TaskDetailsDialog`.

EditReportDialog Stellt einen Texteditor zur Verfügung, um einen Report zu verfassen. Wenn bereits ein Report verfasst wurde, bzw. mit dem Verfassen eines Reports

begonnen wurde, wird automatisch der bereits bestehende Text geladen. Ausserdem wird das Speichern oder Verwerfen des verfassten Reports ermöglicht und somit erfolgt die Navigation zurück zu den *Task*details.

SettingsDialog Stellt Eingabemasken zur Konfiguration von Host/Port des Servers zur Verfügung und zum Setzen eines Passwortes, dass zur Verschlüsselung lokal gespeicherter Daten dient. Die Daten werden in der Settings-Klasse (siehe 6.5.3.5 auf Seite 88 abgespeichert, damit die Klassen, welche die Werte benötigen, darauf zugreifen können. Erlaubt die vorgenommenen Änderungen zu speichern oder zu verwerfen.

PasswordDialog Ermöglicht es ein Passwort einzugeben, bietet einen Listener an, der aufgerufen wird, wenn der Benutzer OK oder Cancel geklickt hat. Die Methode `getResult` gibt das eingegebene Passwort zurück, welches dann mit Hilfe der Settings-Klasse (siehe 6.5.3.5 auf Seite 88) auf Korrektheit geprüft werden kann.

FormFactory Erstellt eine neue Instanz der gewünschten Form, sofern sie noch nicht existiert und gibt sie zurück. Existiert die Form bereits, wird die bestehende Instanz zurückgegeben.

6.5.2.5 Abhängigkeiten von anderen Packages

business.services.WorklistService Wird benötigt, um die Liste des *Tasks* abzurufen.

business.entities.Task Die in dieser Klasse repräsentierten Informationen eines *Tasks* werden vom *GUI* angezeigt. Die Status des *Tasks* können vom *GUI* verändert werden.

business.entities.Report Die in dieser Klasse repräsentierten Daten des Reports werden vom *GUI* angezeigt und Änderungen in diese Klasse zurück gespeichert.

business.Settings Wird benötigt, um die Einstellungen im SettingsDialog zu laden und zu speichern.

6.5.3 Package Business

6.5.3.1 Beschreibung des Package

Dieses Package ist in zwei Subpackages aufgezeigt, das Package Services und das Package Entities. Das Package Services stellt den grössten Teil der Logik zur Verfügung. Es stellt über das Storage-Package eine Verbindung zum *DICOM*-Server her und ruft so Daten vom Server ab und sendet auch Daten an den Server zurück. Das Entities-Package enthält Klassen, welche verschiedene Entitäten des *DICOM/HL7*-Modells darstellen, die grösstenteils rein der Abbildung der Daten dienen.

6.5.3.2 Diagramm

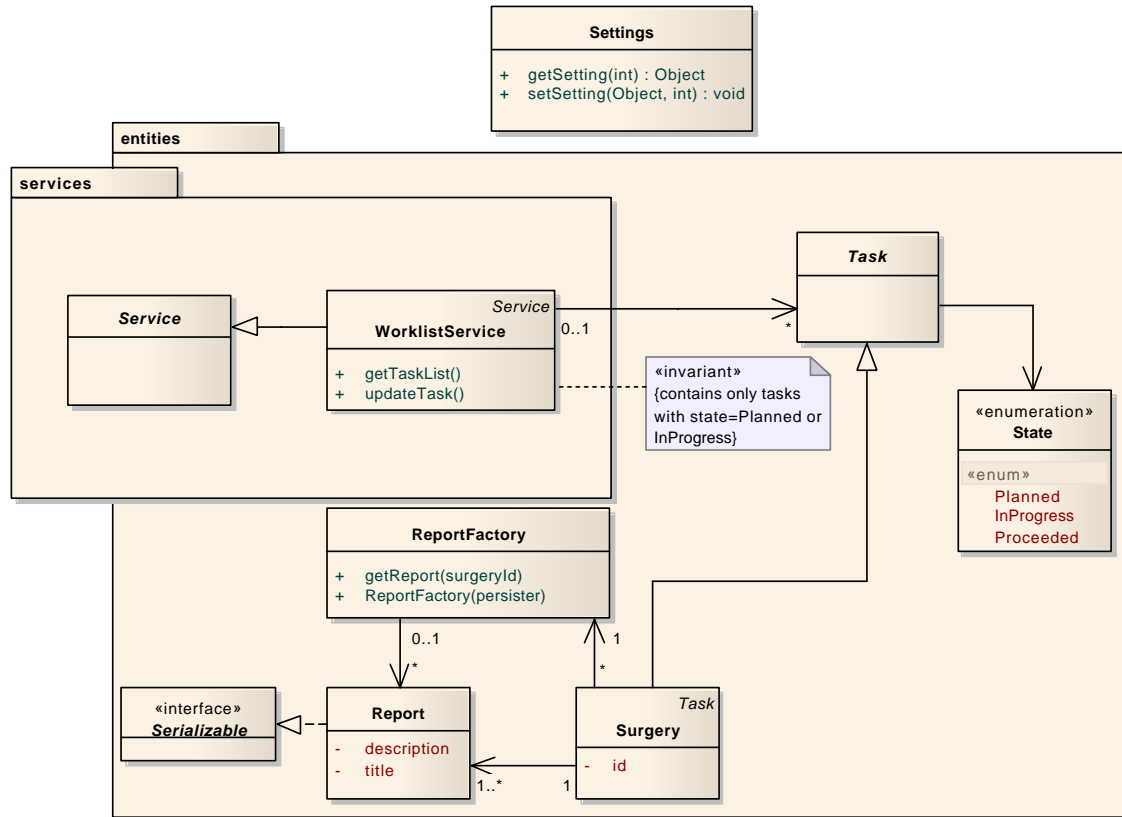


Abbildung (6.4) Klassendiagramm Package Business

6.5.3.3 Schnittstellen

Die Hauptschnittstellen des Business-Layers bestehen aus folgenden Klassen:

WorklistService Wird zum Abrufen von *Tasks* benötigt.

Settings Wird benötigt um Einstellungen zu laden, zu speichern und auf Korrektheit zu überprüfen.

Klassen, die eine Entität darstellen, gehören ebenfalls zur Schnittstelle, da sie vom *WorklistService* geliefert werden, wenn eine *Tasklist* abgerufen wird.

6.5.3.4 Operationen

WorklistService.findTasks Stellt über das Network-Package (siehe 6.5.5 auf Seite 90) eine Verbindung zum *DICOM*-Server her. Ruft dann die IDs aller Patienten vom

Server ab, die den angegebenen Nachnamen haben ¹ und zu einem Task gehören, der zu der angegebenen Modalität gehört. Danach werden alle Patienten-IDs herausgefiltert, deren Vorname, nicht mit dem angegebenen Vornamen übereinstimmt. Zum Schluss werden die eigentlichen Daten vom Server abgerufen und als Suchkriterien die Patienten-IDs und Modalität angegeben.

ReportFactory.persistReport Speichert den Report unter der angegebenen ID (normalerweise die ID des Tasks zu dem der Report gehört) ab. Dazu werden die Daten über mit Hilfe der Kryptographie-Klassen von Bouncycastle verschlüsselt und mit Hilfe der Storage-Klasse persistent abgespeichert.

6.5.3.5 Klassen

WorklistService Ermöglicht das Abrufen von *Tasks* vom Server. Es muss ein Filterkriterium (Patient und/oder Modalität) angegeben werden, nach dem die zurückgegebenen *Tasks* gefiltert werden. Die Filterung selbst wird allerdings bereits auf dem Server vorgenommen, die WorklistService-Klasse leitet die Filterkriterien nur an den Server weiter. Ausserdem ermöglicht es die Klasse, den Status eines *Tasks* auf dem Server zu ändern (inkl. anfügen von Reports an den *Task*). Die Klasse bildet ausserdem die Daten, welche vom Server empfangen werden, auf die Klassen im Entities-Package ab. Für das Abfragen der Daten vom Server wird die Connection-Klasse (siehe 6.5.5.6 auf Seite 93) benötigt, welche die Daten vom Server in einem einheitlichen Format zurückliefert und sie für den Server aufbereitet.

ReportFactory Prüft über die Storage-Klasse (siehe 6.5.4.4 auf Seite 90) des Persistence-Packages, ob es zu einem Task mit einer bestimmten ID bereits einen Report gibt. Wenn ja, wird die gespeicherte Klasse deserialisiert und zurückgegeben. Ansonsten wird eine neue Report-Klasse erstellt und zurückgegeben. Ermöglicht es auch einen Report zu speichern.

Report Stellt die Daten eines «Structured Report» dar und kann sich in den persistenten Speicher serialisieren. Ausserdem bietet die Klasse einen Listener an, der die Listener bei Änderung des Inhalts informiert.

Settings Speichert Einstellungen der Applikation über die Storage-Klasse persistent ab und lädt die Einstellungen von da aus auch wieder in die Applikation. Prüft ausserdem, ob die Werte, welche für die Einstellungen gesetzt sind, korrekt sind. Zudem wird

1 Da die Suche nicht funktioniert, wenn man nach Vor- und Nachnamen sucht, musste die Filterung nach Vornamen auf dem Device vorgenommen werden. Wir vermuten, dass die ein Bug des *DVTk* RIS-Emulators ist, da die Suche nach Nachnamen ja ganz normal funktioniert. Es könnte allerdings auch sein, dass wir die Suche falsch anwenden, das es sich bei der Namenssuche um die Einzige Suche handelt, die nicht standardisiert ist. Leider gibt es aber für den RIS-Emulator kein Conformance-Statement dazu, wie die Suche durchgeführt werden muss.

bei Setzen eines Passwortes ein Schlüssel generiert, der für die Verschlüsselung der Reports benötigt wird. Dieser Schlüssel wird mit Hilfe des Passwortes verschlüsselt abgespeichert. Damit die Korrektheit des Passwortes überprüft werden kann, wird ausserdem noch ein Hash des Passwortes gespeichert.

6.5.3.6 Abhängigkeiten von anderen Packages

persistence.network.Connection Wird benötigt, um mit dem Server zu kommunizieren.

persistence.storage.Storage Wird benötigt, um Daten auf dem persistenten Speicher des mobilen Endgerätes abzuspeichern.

util.crypto.CipheredInputStream Wird benötigt um Reports verschlüsselt abzuspeichern.

util.crypto.CipheredOutputStream Wird benötigt um gespeicherte Reports wieder zu laden.

util.concurrent Wird benötigt um die Netzwerkabfragen asynchron auszuführen.

6.5.4 Package Persistence

6.5.4.1 Beschreibung des Package

Dieses Package ist dafür zuständig, Daten von persistenten Speichern abzurufen bzw. abzuspeichern. Die Daten werden dazu dem persistenten Speicher des mobilen Endgerätes abgespeichert.

6.5.4.2 Diagramm

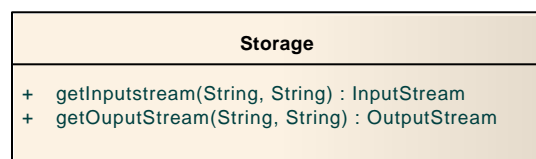


Abbildung (6.5) Klassendiagramm Package Persistence

6.5.4.3 Schnittstellen

Die Klasse **Storage** ist die einzige Schnittstelle dieses Packages, sie stellt Input- und OutputStreams zur Verfügung um die Daten zu Laden und zu speichern.

6.5.4.4 Klassen

Storage Speichert die Daten in einem RecordStore, diese wird von Java *ME* zur Verfügung gestellt und speichert Daten in einer Datenbank ab, die nur von der jeweiligen Anwendung gelesen werden kann. Für das einfache Verarbeiten der Daten wird der RecordStore, der die Daten enthält in einem Input- bzw. OutputStream gekapselt.

6.5.4.5 Abhängigkeiten von anderen Packages

Das Package hat keine Abhängigkeiten zu anderen Packages.

6.5.5 Package Network

6.5.5.1 Beschreibung des Package

Dieses Package kapselt die *DICOM* spezifische Netzwerkfunktionalität. Dies beinhaltet einerseits das Assoziieren mit einem Server, als auch das Abfragen von Worklist-Einträgen. Das Package ist mit einem Präfix «dicom» abgegrenzt, um dies als standardspezifische Implementierung zu kennzeichnen. Das Network Package beinhaltet diverse Subpackages, welche einzelne vom Standard vorgeschriebene Funktionalitäten implementieren. Dies sind einerseits Werte-Repräsentation, die Darstellung der Nachrichten und das Parsing der empfangenen Nachrichten. Die einzelnen Packages werden im weiteren Verlauf dieses Dokuments genauer beschrieben. Um der Komplexität gerecht zu werden, sind ausführliche Erklärungen mit Verweis auf den *DICOM*-Standard an den entsprechenden Stellen vorgesehen.

6.5.5.2 Diagramm

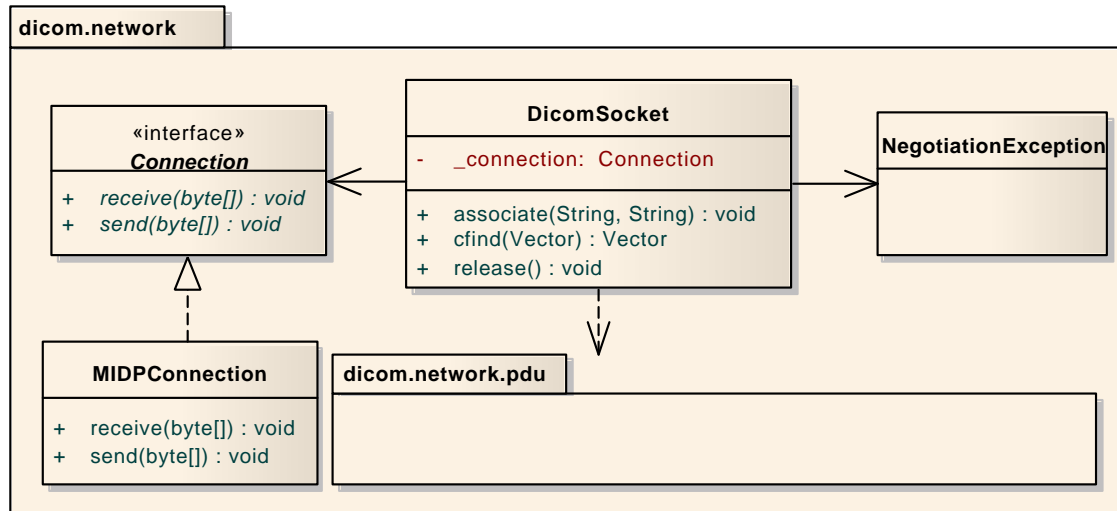


Abbildung (6.6) Klassendiagramm Package Network

6.5.5.3 Schnittstellen

Um die Netzwerkschicht zu benutzen, gibt es eine Hauptschnittstelle, welche durch die Klasse «DicomSocket» dargestellt wird. Diese Klasse bietet drei Hauptfunktionen an:

- Association Establish
- Composite Find
- Association Release

Für den logischen Ablauf dieser drei Funktionen (siehe Kapitel 5.8 auf Seite 72). Den Aufbau der Nachrichten, welche durch diese Operationen verschickt werden, sind unter 6.5.6 auf Seite 94 einsehbar. Die Klasse «DicomSocket» nimmt als Konstruktor Parameter eine «Connection» entgegen. Dies ermöglicht ein transparentes Abbilden von verschiedenen Verbindungsmöglichkeiten (z. B. wäre eine *USB*¹-Schnittstelle so auch denkbar). Diese Verbindungsimplementation muss lediglich das Interface «Connection» implementieren.

6.5.5.4 Operationen

associate Diese Funktion stellt eine Verbindung mit dem *DICOM*-Server her. Es müssen zwei Parameter mitgegeben werden, einerseits der Name der «Calling-Entity» und

¹ *Universal Serial Bus*

andererseits dieser der «Called-Entity». Die «Calling-Entity» ist der lokale Name der Applikation. Die «Called-Entity» ist der Name des Server-Dienstes, unter welchem der Worklist-Dienst läuft. Ein Associate handelt mit dem Server einen Transfersyntax aus (siehe 6.5.5.5). Ausserdem wird überprüft, ob die Protokollversion des Servers mit dieser der Implementation übereinstimmt (Die Protokollversion ist momentan 1). Sollte die Protokollversion nicht übereinstimmen, wird eine «ProtocolVersionMismatchException» geworfen. Sollte kein gemeinsamer Transfersyntax unterstützt sein, wird eine «NegotiationException» geworfen. Zusätzlich ist es möglich, dass eine «ValueRepresentationException» auftritt. Auf diese wird genauer in 6.5.7 auf Seite 105 eingegangen.

cfind Die `cfind` (Composite-Find) Funktion stellt eine Anfrage für bestimmte Daten an den Server. Diese Anfrage folgt speziellen Regeln: Es müssen Attribute definiert werden, nach denen gesucht wird. Diese werden leer mitgeschickt. Zusätzlich können optionale Attribute definiert werden, welche ein Suchkriterium enthalten. Diese Attribute werden «Presentation Data Values» genannt (siehe 6.5.6.3 auf Seite 96). Die `cfind` Operation liefert alle gefundenen Attribute vom Server zurück. Zu beachten ist, dass die Struktur der Attribute der Anfrage korrekt sein muss, d. h. sie müssen in der korrekten Hierarchie übermittelt werden (siehe dazu 6.5.6.4 auf Seite 96).

release Die `RELEASE` Operation terminiert lediglich die Verbindung mit dem *DICOM*-Server. Es wird hierbei nur eine Nachricht mit einer vorgeschriebenen Nummer übertragen, ohne weiteren Dateninhalt.

6.5.5.5 Transfersyntax

Ein Transfersyntax kann als Übertragungssprache angesehen werden. Wir haben während der Implementation drei Hauptsyntaxen angetroffen:

- Implicit Little Endian
- Explicit Little Endian
- Explicit Big Endian

Transfersyntaxen unterscheiden sich durch die Datendarstellung und die Codierung der Längenfelder. Da in dieser Arbeit nur der Transfersyntax «Implicit Little Endian» verwirklicht wurde, wird nur auf diesen genauer eingegangen (siehe [Gla06]). Generell kann man sagen, dass der Unterschied zwischen Implicit und Explicit darin besteht, dass bei Längenangaben, die Explicit Syntaxen eine Codierung der Wertart vorschreibt, was bei Implicit nicht der Fall ist (siehe Beispiel). Little Endian setzt voraus, dass die niederwertigsten Bytes zuerst geschrieben werden. Dies ist aber auch noch abhängig von den Werte-Repräsentationen (siehe 6.5.7 auf Seite 105).

Beispiel Implicit (zwecks Verständlichkeit Big-Endian):

<i>Group-ID</i>	<i>Element-ID</i>	<i>Length</i>	<i>Value</i>
0x0000	0x0900	0x00000002	0x0000

Tabelle (6.1) Beispiel Implicit Codierung

Dies wäre eine Status Presentation Data Value, welche aussagt, dass keine weiteren Nachrichten vorhanden sind. Beispiel Explicit (zwecks Verständlichkeit Big-Endian):

<i>Group-ID</i>	<i>Element-ID</i>	<i>Value-Code</i>	<i>Length</i>	<i>Value</i>
0x0000	0x0900	US	0x0002	0x0000

Tabelle (6.2) Beispiel Explicit Codierung

Dies wäre eine Status Presentation Data Value, welche aussagt, dass keine weiteren Nachrichten vorhanden sind. Die Werte-Codierung ist «Unsigned Short» (Status PDV: [Nat08c, Tabelle 9.3-4], Werte-Repräsentation US: [Nat08a, Seite 30]).

6.5.5.6 Klassen

Connection Das Connection Interface soll verschiedene Arten von Verbindungen abstrahieren, damit es möglich ist z. B., über eine *USB*-Verbindung die *DICOM* Netzwerkfunktionalität auszuüben.

MIDPConnection Spezifische Implementierung des Connection Interfaces für eine WLAN Verbindung mittels eines mobilen Gerätes.

DicomSocket Die Klasse DicomSocket kapselt die *DICOM* spezifische Funktionalität dieser Applikation, d. h. Association, Composite-Find und Release. Ausserdem wird die Aushandlung des Transfersyntaxes und die Überprüfung der Protokollversion darin sichergestellt.

NegotiationException Die NegotiationException wird geworfen, wenn der vom Client gewählte Transfersyntax nicht vom Server unterstützt wird. Sollte dies der Fall sein, ist keine Kommunikation mehr möglich.

6.5.5.7 Abhängigkeiten von anderen Packages

dicom.network.pdu *DICOM*-Nachrichten für die Association und Composite-Find (siehe 6.5.6 auf der nächsten Seite).

dicom.network.vr Werte-Repräsentationen von zu Übertragenden und zu Empfangenden «Presentation Data Values» sowie das Encoding/Decoding anhand des gesetzten Transfersyntaxes (siehe 6.5.6.3 auf Seite 96).

dicom.util Spezielle Streams für *DICOM* spezifische Funktionalität.

6.5.6 Package PDU

6.5.6.1 Beschreibung des Package

Dieses Package beinhaltet die Implementation der *DICOM*-Nachrichten und die darin enthaltenen Felder. Da die Struktur durch den Standard fix definiert ist, wurde auch die Hierarchie der Nachrichten und Feldern fest definiert. Allgemein betrachtet hat eine Nachricht immer einen Typ und eine Länge und kann mehrere Felder enthalten. Einige Felder können Unterfelder enthalten und fließen somit in die Längenangaben einzelner Felder mit ein. Ebenfalls in diesem Package enthalten ist die Implementation der «Presentation Data Values» (siehe 6.5.6.3 auf Seite 96) sowie der «Information Object Definition» (siehe 6.5.6.4 auf Seite 96).

6.5.6.2 Diagramm

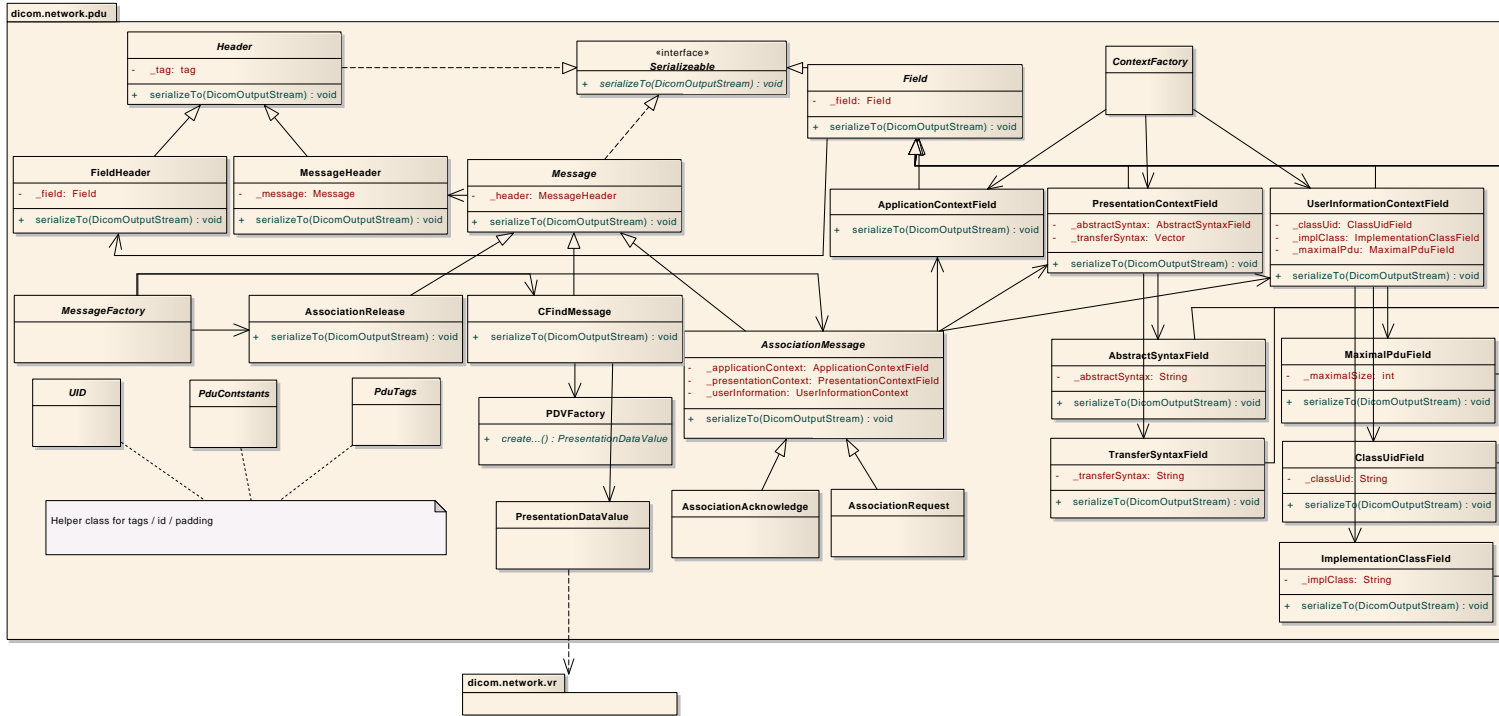


Abbildung (6.7) Klassendiagramm Package PDU

6.5.6.3 Presentation Data Value

Ein «Presentation Data Value» (nachfolgend PDV) ist eine Kombination von einer Werte-Repräsentation und einer «Information Object Definition». Im *DICOM* Kontext bezogen auf diese Implementierung bedeutet dies, dass eine PDV die Gruppierung dieser Funktionalitäten darstellt (für Werte-Repräsentationen siehe 6.5.7 auf Seite 105).

6.5.6.4 Information Object Definition

Eine «Information Object Definition» (nachfolgend IOD) ist eine Zuordnung von Identifikationsnummern zu einer Werte-Repräsentation. Diese Identifikationsnummern sind zwei Byte grosse Nummern, aufgeteilt in Group-ID und Element-ID. Jedem Paar dieser Nummern ist eine Werte-Repräsentation zugeordnet, welche definiert, wie der Wert dieser IOD dargestellt wird. Eine IOD ist also allgemein betrachtet ein Datenfeld, wie beispielsweise der Name eines Patienten oder seine Allergien. Eine genaue Auflistung aller IODs und deren Werte-Repräsentation ist unter [Nat08b, Seite 8ff] zu finden.

6.5.6.5 Schnittstellen

Die Schnittstellen dieses Packages stellen die folgenden beiden Klassen dar:

- `MessageFactory`
- `PDVFactory`

Die `MessageFactory` kapselt das Erstellen verschiedener Nachrichten und setzt sie anhand von Parametern oder der vom Standard vordefinierten Struktur zusammen, so dass sie nur noch über die Netzwerkschnittstelle verschickt werden müssen. Die `PDVFactory` nimmt die Aufgabe wahr, für gegebene IOD-Nummern die entsprechende Werte-Repräsentation aus einem Konfigurationsfile zu lesen und den mitgegebenen Wert zu setzen (die Überprüfung des Wertes wird von der Werte-Repräsentation übernommen).

6.5.6.6 Operationen

`MessageFactory.createMwlAssociationRequest` Erstellt eine Request-Nachricht für eine Association zu einen *DICOM*-Server. Dieser kann dann serialisiert werden und über die Netzwerkverbindung verschickt werden.

`MessageFactory.createMwlAssociationResponse` Nimmt die empfangen Daten nach einem Association Request, um diese in die vom Server erhaltene Antwort zu parsen.

`MessageFactory.createCFindRequestCommand` Erstellt eine Composite-Find Command Request Nachricht. Ein Command Request wird immer vor einem Data Request abgeschickt und beinhaltet Informationen, welche vom Server benötigt werden, um die Composite-Find Operation durchzuführen.

MessageFactory.createCFindRequestData Erstellt eine Composite-Find Data Request Nachricht. Diese enthält Suchattribute sowie solche, die vom Server mit Werten gefüllt werden sollen.

MessageFactory.createCFindResponseData Nimmt die empfangenen Daten nach einem Composite-Find Request, um diese in die vom Server erhaltene Antwort zu parsen.

MessageFactory.createAssociationRelease Erstellt eine Association Release Nachricht, um die Verbindung mit dem *DICOM*-Server zu beenden.

PDVFactory.setDecoder / **PDVFactory.setEncoder** Setzt den durch den Transfer-syntax bestimmten Encoder/Decoder, welcher für die «Presentation Data Values» notwendig ist.

PDVFactory.create Erstellt eine PresentationDataValue (PDV). Es gibt vier Versionen dieser Methode mit unterschiedlichen Parametern. Die erste erstellt eine PDV anhand von Group-ID, Element-ID und einem Wert. Diese wird benötigt, um PDVs mit Suchkriterien zu übertragen. Die zweite Version nimmt einen InputStream entgegen, welcher erhaltene Daten von einem Server enthält. Es wird dann eine PDV aus dem Stream geparsed. Nummer drei erstellt eine PDV anhand von Group-ID, Element-ID und einem InputStream. Diese Variante ist notwendig, um spezielle Werte-Repräsentationen, welche Unterfelder enthalten, zu parsen (siehe 6.5.7.5 auf Seite 108). Die letzte Variante erstellt eine leere PDV, nimmt also nur eine Group-ID und eine Element-ID entgegen.

6.5.6.7 Klassen

Header Abstraktion eines Nachrichten- oder Feldheaders. Ein *DICOM*-Header besteht immer aus einem Typ und einer Länge, getrennt von einem Padding-Byte mit dem Wert 0x00. Die Länge ist abhängig davon, ob es sich um ein Feld oder einen Nachrichten Header handelt, zwei bzw. vier Bytes gross. Deshalb gibt es eine Klasse FieldHeader und MessageHeader.

FieldHeader Die Klasse FieldHeader dient lediglich dazu, das Schreiben der Datenlänge zu übernehmen. Als Konstruktorparameter wird das zugehörige Feld übergeben, welches eine «length» Methode implementiert. So kann beim Serialisieren der Nachricht die Länge automatisch ermittelt werden (ein direktes setzen der Länge beim Konstruieren der Klasse ist nicht möglich, da die Datenlänge erst beim Serialisieren exakt fest steht).

MessageHeader Ist ein Ebenbild von FieldHeader, mit dem Unterschied, dass die Länge nicht mit zwei sondern mit vier Bytes dargestellt wird und im Konstruktor eine Message erwartet wird, welche ebenfalls wie ein Feld eine «length» Methode implementiert.

Field Die Klasse Field bildet die Basis aller Felder, welche in *DICOM*-Nachrichten enthalten sein können. Ein Feld stellt demzufolge sicher, dass Subklassen eine «length» Methode implementieren und erstellt den FieldHeader für dieses Feld (für die Berechnung von Feldlängen siehe 6.5.6.8 auf Seite 104).

Message Die Klasse Message bildet die Basis aller Nachrichten, welche verschickt werden können. Eine Nachricht stellt demzufolge ähnlich wie ein Feld die Implementation einer «length» Methode durch Subklassen sicher und verwaltet den Nachrichten-Header (für die Berechnung von Nachrichtenlängen siehe 6.5.6.8 auf Seite 104).

AbstractSyntaxField Eine konkrete Field Subklasse. Ein AbstractSyntaxField wird bei einer Association mitgeschickt und ist eine normierte ID. Diese ID beschreibt die Art der Aktion, welche über diese Association durchgeführt werden soll. Dies ist in diesem Falle z. B. eine Worklist Query. Das AbstractSyntaxField ist ein Unterfeld des PresentationContextFields.

Byte	Field Name	Description
0	Item Type	0x30
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-4	Item Length	Feldlänge
5-xxx	Abstract Syntax ID	Gültige Abstract Syntax ID (z. B. Worklist Query)

Tabelle (6.3) Abstract Syntax Field (siehe auch [Nat08d, Tabelle 9-14])

ApplicationContextField Eine konkrete Field Subklasse. Ein ApplicationContextField wird bei einer Association mitgeschickt und enthält eine fix definierte Identifikationsnummer. Der Application Context beschreibt eigentlich nur, dass es sich um *DICOM* handelt.

Byte	Field Name	Description
0	Item Type	0x10
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-4	Item Length	Feldlänge
5-xxx	Abstract Syntax ID	Gültige Application Context ID

Tabelle (6.4) Application Context Field (siehe auch [Nat08d, Tabelle 9-12])

ClassUidField Eine konkrete Field Subklasse. Ein ClassUidField wird bei einer Association mitgeschickt und enthält die Implementation Class als ID. ClassUidField ist ein Unterfeld von UserInformationContextField.

Byte	Field Name	Description
0	Item Type	0x52
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-4	Item Length	Feldlänge
5-xxx	Class ID	Gültige Class ID

Tabelle (6.5) Class UID Field (Siehe auch [Zei] - Referenz Library)

ImplementationClassField Ähnlich dem ClassUidField, jedoch mit einem Implementationsnamen und keiner ID. ImplementationClassField ist ein Unterfeld von UserInformationContextField.

Byte	Field Name	Description
0	Item Type	0x55
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-4	Item Length	Feldlänge
5-xxx	Implementation Name	Implementation Name

Tabelle (6.6) Implementation Class Field (Siehe auch [Zei] - Referenz Library)

MaximalPduField Eine konkrete Field Subklasse. Ein MaximalPduField wird während einer Association mitgeschickt und setzt die maximale Grösse für Composite-Find Nachrichten (bzw. P-Data-TF, siehe 6.5.6.9 auf Seite 104). MaximalPduField ist ein Unterfeld vom UserInformationContextField.

Byte	Field Name	Description
0	Item Type	0x55
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-4	Item Length	Feldlänge (Fix 4)
5-8	Maximal Size	Maximale länge für Nachrichten

Tabelle (6.7) Maximal Pdu Field (siehe auch [Nat08d, Tabelle D.1-1])

PresentationContextField Eine konkrete Field Subklasse. Ein PresentationContextField wird während einer Association mitgeschickt und enthält die Transfersyntax Felder, welche ausgehandelt werden müssen.

Byte	Field Name	Description
0	Item Type	0x20
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-4	Item Length	Feldlänge
5	Context ID	Context ID (1-255)
6-8	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
9-xxx	Transfer Syntax	Transfer Syntax Unterfelder

Tabelle (6.8) Presentation Context Field Request (siehe auch [Nat08d, Tabelle 9-13])

Byte	Field Name	Description
0	Item Type	0x21
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-4	Item Length	Feldlänge
5	Context ID	Context ID (1-255)
6	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
7	Result	Resultat Identifikation (Accept, Reject,...)
8	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
9-xxx	Transfer Syntax	Transfer Syntax Unterfelder

Tabelle (6.9) Presentation Context Field Response (siehe auch [Nat08d, Tabelle 9-18])

TransferSyntaxField Eine konkrete Field Subklasse. Ein TransferSyntaxField wird während einer Association mitgeschickt und enthält die Transfersyntax ID's. TransferSyntaxField ist ein Unterfeld von PresentationContextField.

Byte	Field Name	Description
0	Item Type	0x40
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-4	Item Length	Feldlänge
5-xxx	Transfer Syntax	Transfer Syntax ID

Tabelle (6.10) Transfer Syntax Field (siehe auch [Nat08d, Tabelle 9-15])

UserInformationContextField Eine konkrete Field Subklasse. Ein UserInformationContextField wird während einer Association mitgeschickt und enthält das MaximalPduField, ImplementationClassField und ClassUidField.

Byte	Field Name	Description
0	Item Type	0x50
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-4	Item Length	Feldlänge
5-xxx	Unterfelder	MaximalPduField, ImplementationClassField und ClassUidField

Tabelle (6.11) User Information Context Field (siehe auch [Nat08d, Tabelle 9-16])

ContextFactory Factory-Klasse um Context Felder zu erstellen (Felder mit Unterfeldern). Dies beinhaltet das Erstellen von ApplicationContextField, UserInformationContextField und PresentationContextField.

AssociationMessage Basisklasse für AssociationRequest und AssociationResponse. Kapselt die , welche in einer Association vorhanden sind (ApplicationContextField, UserInformationContextField und PresentationContextField, genereller Aufbau siehe [Nat08d, Seite 33]).

AssociationRequest Subklasse von AssociationMessage. Stellt eine Association Anfrage an den *DICOM*-Server.

Byte	Field Name	Description
0	Item Type	0x01
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-6	Message Length	Nachrichtenlänge
7-8	Protocol Version	Protokoll Version (1)
9-10	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
11-26	Called Entity	Name des Server-Dienstes
27-42	Calling Entity	Name des Clients
43-74	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
75-xxx	Fields	Felder (ApplicationContextField, UserInformationContextField und PresentationContextField)

Tabelle (6.12) Association Nachricht (siehe auch [Nat08d, Tabelle 9-11])

AssociationResponse Subklasse von AssociationMessage. Stellt die Antwort auf eine Association von einem *DICOM*-Server dar.

Byte	Field Name	Description
0	Item Type	0x02
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-6	Message Length	Nachrichtenlänge
7-8	Protocol Version	Protokoll Version (1)
9-10	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
11-26	Called Entity	Name des Server-Dienstes (Sollte beim Empfangen nicht getestet werden)
27-42	Calling Entity	Name des Clients (Sollte beim Empfangen nicht getestet werden)
43-74	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
75-xxx	Fields	Felder (ApplicationContextField, UserInformationContextField und PresentationContextField)

Tabelle (6.13) Association Response (siehe auch [Nat08d, Tabelle 9-17])

AssociationRelease Nachricht, um eine Verbindung mit dem *DICOM*-Server zu beenden.

Byte	Field Name	Description
0	Item Type	0x05
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-6	Message Length	Nachrichtenlänge (Fix 4)
7-10	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)

Tabelle (6.14) Association Release (siehe auch [Nat08d, Tabelle 9-24])

CFindMessage Kapselt die Funktionalität einer Composite-Find Data/Command Request/Response. Eine CFindMessage beinhaltet mehrere PresentationDataValues, welche der Datenselektion dienen (Data Request/Response) oder Informationen über die Abfrage enthalten (Command Request/Response). Ob es sich um eine Data oder Command CFindMessage handelt wird anhand eines Status-Flags entschieden. Dies kann vier verschiedene Werte annehmen:

- Bit 0 = 0: Data Message
- Bit 0 = 1: Command Message
- Bit 1 = 0: Not Last Fragment
- Bit 1 = 1: Last Fragment

Demzufolge kann das Status-Flag 0x00, 0x01, 0x02 oder 0x03 annehmen. Last-Fragment bezieht sich nicht darauf, ob es noch mehr Nachrichten gibt, sondern ob diese Nachricht komplett ist, da die Fragmentierung von Nachrichten auf der *DICOM*-Protokollebene selber geschieht und nicht auf tieferen Netzwerkschichten (die Fragmentierung geschieht, sobald die Nachrichtenlänge oberhalb der Maximal

*PDU*¹ Size ist, welche während der Association ausgehandelt wird). Ob es sich um die letzte Nachricht handelt, wird anhand einer Status Presentation Data Value signalisiert. Ist der Wert dieses Status 0x00, handelt es sich um die letzte Nachricht. Diese Status PDV ist nur in Command CFindMessages enthalten. Dies bedeutet auch, dass eine Response vom Server immer aus Paaren von Command und Data Nachrichten besteht. Zuerst wird immer eine Command Nachricht geschickt, gefolgt von einer Data Nachricht, ausser es handelt sich um die letzte Nachricht, dann wird nur noch eine Command Nachricht übertragen (demzufolge hat eine Composite-Find Response n Data und n + 1 Command Nachrichten).

Byte	Field Name	Description
0	Item Type	0x04
1	Padding	0x00 (Sollte beim Empfangen nicht getestet werden)
3-6	Message Length	Nachrichtenlänge
7-10	PDV Length	Länge aller PDV's zusammen
11	Context ID	ID des während der Association gewählten Presentation Contexts
12	Command / Data / Fragment	0x00, 0x01, 0x02 oder 0x03 (Command/Data/Fragment)
13-xxx	Presentation Data Values	Alle Presentation Data Values

Tabelle (6.15) Presentation Data Transfer (siehe auch [Nat08d, Tabelle 9-22 und 9-23])

PresentationDataValue Klasse, welche die Informationen Object Definitions (siehe 6.5.6.4 auf Seite 96) und die Werte-Repräsentationen kapselt (siehe 6.5.7 auf Seite 105). Ebenfalls verwaltet die PresentationDataValue Klasse die Encoder/Decoder, welche durch den Transfersyntax (6.5.5.5 auf Seite 92) bestimmt werden (logische Funktionalität von Presentation Data Values siehe 6.5.6.3 auf Seite 96).

Byte	Field Name	Description
0-1	Group ID	Group ID ([Nat08b])
2-3	Element ID	Element ID ([Nat08b])
4-7	PDV Length	Länge der PDV
8-xxx	Value Representation	Wert anhand einer Wert-Representation (6.5.7 auf Seite 105)

Tabelle (6.16) Presentation Data Value (siehe auch [Nat08d, Tabelle 9-22 und 9-23])

MessageFactory Factory Klasse um Association- und CFindMessages zu erstellen. Da gewisse Werte enthalten sein müssen oder zum voraus definiert werden können, wird dies direkt in den einzelnen Methoden automatisch durchgeführt.

¹ *Protocol Data Unit*

PDVFactory Die PDVFactory erstellt eine PresentationDataValue anhand der richtigen Werte-Repräsentation, welche durch die Group- und Element-ID abhängig ist. Diese Zuordnung wird mit Hilfe der Klasse IODProperties aus einem Konfigurationsfile gelesen.

IODProperties Die Klasse IODProperties liest aus einem Konfigurationsfile (InformationObjectDefinition.properties) die registrierten Group- und Element-ID's mit zugehöriger Werte-Repräsentation (Information Object Definition: 6.5.6.4 auf Seite 96).

UID Die Klasse UID enthält spezielle Konstanten, sogenannte Unique Identifiers. Ein Unique Identifier ist eine mit «.» (Punkt) getrennte Nummer (z. B. 1.2.10008.1.2). Es gibt UIDs für die Transfersyntaxen, Abstractsyntax, Class-UID etc.

PduTags Die Klasse PduTags enthält Konstanten, welche die Feld- und Nachrichtentypen beschreiben. Beispielsweise eine AssociationMessage für ein Association Request hat den Typ 0x01.

PduConstants Die Klasse PduConstants enthält konstante Byte-Werte, welche oft in Nachrichten gebraucht werden (wie beispielsweise 0x00 oder 0x20 als Padding).

6.5.6.8 Berechnung von Feld- und Nachrichtenlängen

Die Länge eines Feldes bzw. einer Nachricht beinhaltet immer die Länge ab dem Längensfeld bis zum Ende des Feldes bzw. der Nachricht. Dies bedeutet, dass der Header selber nicht in die Nachrichtenlänge mit aufgenommen wird. Sollte ein Feld Unterfelder enthalten, werden diese komplett in die Länge aufgenommen, d. h. inklusive Headerlänge. Man kann also generell sagen, es geschieht abhängig vom «Ort» an dem man sich momentan in der Nachricht oder im Feld befindet, eine unterschiedliche Längenberechnung.

6.5.6.9 P-Data-TF

P-Data-TF steht für Presentation Data Transfer und ist eine vom Standard vorgeschriebene Art, Daten zu übertragen. Die CFindMessage ist gemäss dieses Protokolls aufgebaut. Dem DICOM-Server wird mittels einer Presentation Data Value im Datenteil der CFindMessage signalisiert, dass es sich um eine Composite-Find Nachricht handelt.

6.5.6.10 Abhängigkeiten von anderen Packages

dicom.network.vr Werte-Repräsentation für die PresentationDataValues und Encoder/Decoder der Transfersyntaxen.

dicom.network.pdu.parser Nachrichten- und Feldparser für das Erstellen von empfangenen Nachrichten.

dicom.util Spezielle Streams für *DICOM* spezifische Funktionalität.

6.5.7 Package VR

6.5.7.1 Beschreibung des Package

Das Package VR (Value Representation) enthält alle notwendigen Werte-Repräsentation, welche im *DICOM*-Standard festgehalten sind (siehe [Nat08a, Tabelle 6.2-1]). Diese Werte-Repräsentation behandeln unterschiedliche Datentypen und deren Darstellung (bzw. ihr Format). Ausserdem in diesem Package enthalten sind der Encoder und Decoder für den Transfersyntax (siehe 6.5.5.5 auf Seite 92) «Implicit Little Endian» (siehe [Gla06]).

6.5.7.2 Diagramm

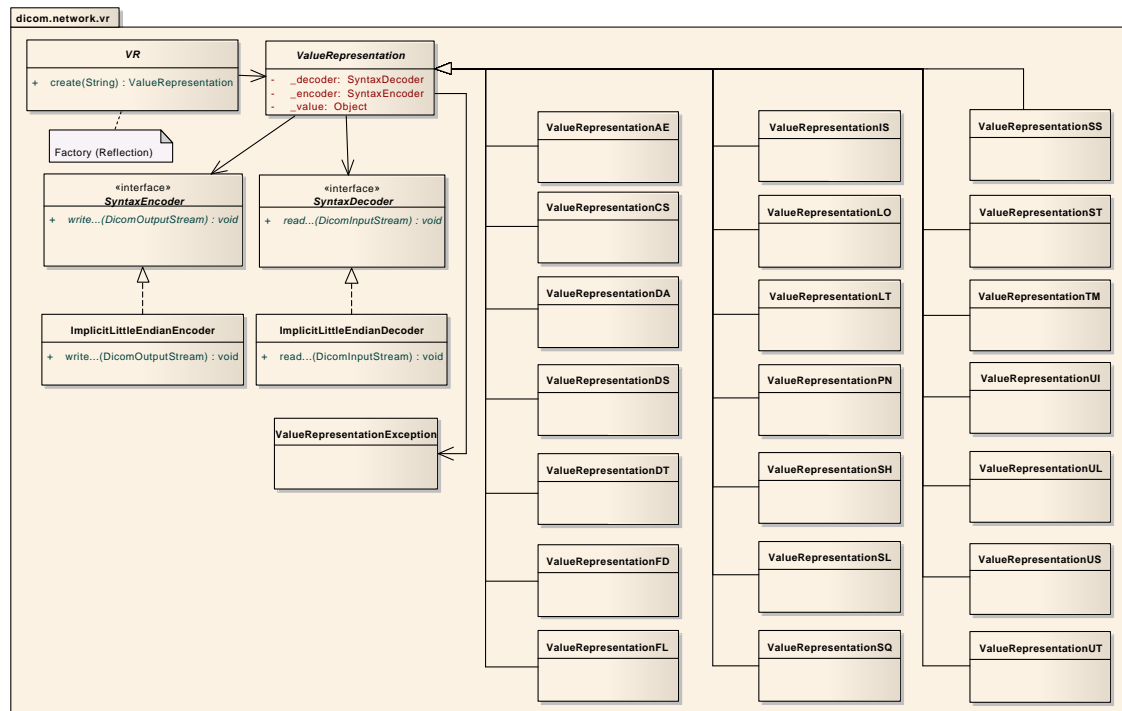


Abbildung (6.8) Klassendiagramm Package VR

6.5.7.3 Schnittstellen

Alle Werte-Repräsentationen im Package folgen einem konkreten Namens-Schema:

- ValueRepresentation + «Value-Code»

Der «Value-Code» wird immer durch zwei Zeichen dargestellt und ist in der Datei `InformationObjectDefinition.properties` einer Information Object Definition zugeordnet (siehe 6.5.6.4 auf Seite 96 bzw. 6.5.6.7 auf Seite 104). Dieser «Value-Code» ist definiert in [Nat08a, Tabelle 6.2-1] bzw. [Nat08b, Seite 8ff] als Zuordnung zu einer Group-ID und Element-ID. Die Schnittstelle wird durch die Klasse «VR» dargestellt, welche eine Factory-Methode «create» enthält. Diese Methode kann mit einem «Value-Code» parametrisiert werden (gelesen durch die Klasse «IODProperties» in `dicom.network.pdu`) und gibt eine Instanz dieser Werte-Repräsentation, erstellt durch Reflection, zurück.

6.5.7.4 Operationen

VR.create Erstellt eine `ValueRepresentationXX` anhand des «Value-Codes», beschrieben in [Nat08a, Tabelle 6.2-1]

6.5.7.5 Klassen

ValueRepresentation Abstrakte Basisklasse für alle ValueRepresentations. Zwingt ableitende Klassen zur Implementierung von Methoden, um die maximale Wertelänge zu bestimmen oder den Wert auf einen `OutputStream` zu schreiben. Da die Datentypen je nach ValueRepresentation unterschiedlich sind, ist die Methode `setValue` mit einem `Object` parametrisiert (keine Generics, da Compiler Compliance 1.4). Eine spezifische ValueRepresentation weiss jedoch, was sie für einen Wert erwartet und kann deshalb den entsprechenden Cast bzw. die Interpretierung des Wertes korrekt und unabhängig ausführen. Ausserdem sind diverse Überprüfungsmechanismen für die korrekte Wertedarstellung und Interpretation in diese Basisklasse ausgelagert, so dass sie von unterschiedlichen ValueRepresentations genutzt werden können. Eine Liste aller ValueRepresentations ist unter [Nat08a, Tabelle 6.2-1] zu finden. Alle Subklassen von ValueRepresentation entsprechen einer der aufgelisteten ValueRepresentation in dieser Tabelle.

ValueRepresentationAE ValueRepresentation für Application Entity Namen. Kann Leading und Trailing Leerschläge (0x20) enthalten. Nicht erlaubte Zeichen: 5CH (Backslash) und Kontrollzeichen LF, FF, CR und ESC. Die maximale Länge beträgt 16 Bytes.

ValueRepresentationCS ValueRepresentation für Code String. Code Strings werden beispielsweise für «Smoking Status» verwendet und bestehen aus Grossbuchstaben, «_» (Underscore) und Zahlen von 0–9. Kann wie Application Entity Leading und Trailing Leerschläge (0x20) enthalten. Die maximale Länge beträgt 16 Bytes.

ValueRepresentationDA ValueRepresentation für Date. Ist ein String mit dem Format YYYYMMDD (Jahr, Monat, Tag). Zahlen von 0-9 sind erlaubt, die Länge ist 8 Bytes.

ValueRepresentationDS ValueRepresentation für Decimal String. «Fixed-Point» oder «Floating-Point». Exponenten können mit E oder e dargestellt werden (Scientific Representation). Erlaubt sind Zahlen von 0–9 mit einem «.» (Punkt) als Dezimaltrennzeichen. Optional kann ein + oder – am Anfang des Wertes angegeben werden. Die maximale Länge beträgt 16 Bytes.

ValueRepresentationDT ValueRepresentation für Date Time. Date Time hat das Format YYYYMMDDHHMMSS.FFFFFFFF&ZZXX (Jahr, Monat, Tag, Stunde, Minute, Sekunde, «Fractional», Zeitzone). «Fractional» ist der genaue Sekundenanteil (z. B. 0.001000 ist eine Millisekunde). Die Zeitzone ist definiert als &=+/-, ZZ=Stunden, XX=Minuten im Verhältnis zu UTC. Der «Fractional» Teil kann weggelassen werden, dies bedeutet aber auch, dass keine Zeitzone angegeben werden kann. Die maximale Länge beträgt 26 Bytes.

ValueRepresentationFD ValueRepresentation für Floating Point Double. Double im IEEE754:1985 Format ([HDXY]). Acht Byte fixe Grösse.

ValueRepresentationFL ValueRepresentation für Floating Point Single. Float im IEEE754:1985 Format ([HDXY]). Vier Byte fixe Grösse.

ValueRepresentationIS ValueRepresentation für Integer String. Integer als String zur Basis 10. Enthält Zeichen 0–9, optional +=-. Darf Leading oder Trailing Leerschläge (0x20) enthalten. Range von und mit -2^{31} bis und mit $2^{31} - 1$. Maximale Länge beträgt 12 Bytes

ValueRepresentationLO ValueRepresentation für Long String. Zeichenkette mit Leading oder Trailing Leerschlägen (0x20). Kein «\» (Backslash) und kein ESC. Maximal 64 Bytes Länge.

ValueRepresentationLT ValueRepresentation für Long Text. Zeichenkette, welche Kontrollzeichen wie CR, LF, FF und ESC enthalten kann. «\» (Backslash) erlaubt. 10240 Bytes Maximum.

ValueRepresentationPN ValueRepresentation für Person Name: «Family Name Complex^Given Name Complex^Middle Name^Name Prefix^Name Suffix». Einzelne Werte in dieser Kette können weggelassen werden. Sollten ab einem Punkt alle Werte bis zum Ende fehlen, können die Trennzeichen am Ende entfallen. Maximale Länge beträgt 64 Bytes. Beispiel:

- Rev. John Robert Quincy Adams, B.A. M.Div. = Adams^John Robert Quincy^^Rev.^B.A. M.Div.

ValueRepresentationSH ValueRepresentation für Short String. Zeichenkette mit Leading oder Trailing Leerschlägen (0x20). Kein «\» (Backslash) und keine Kontrollzeichen, ausser ESC. Die maximale Länge beträgt 16 Bytes.

ValueRepresentationSL ValueRepresentation für Signed Long. vorzeichenbehafteter Integer im 2er Komplement. Range von und mit -2^{31} bis und mit $2^{31} - 1$. Die Länge ist fix vier Bytes.

ValueRepresentationSQ ValueRepresentation für Sequence. Die Sequence ist eine spezielle Art der Wertecodierung, da sie als einzige Subdaten enthalten kann (dies wird als Item bzw. Folderstruktur bezeichnet). Der Sequence ist im Standarddokument eine Teilsektion gewidmet (siehe [Nat08a, Sektion 7.5]). Eine Sequence kann als Länge eine definierte oder undefinierte Länge haben. Speziell zu beachten ist hier lediglich die undefinierte Länge (bei definierter Länge werden die Subdaten direkt gelesen). Eine Sequence enthält ein- oder mehrere Subdatenelemente, dargestellt als Presentation Data Value (siehe 6.5.6.3 auf Seite 96). Eine Sequence beginnt mit einer Presentation Data Value, welche die Group-ID=0xFFFFE und die Element-ID=0xE000 hat. Dies ist ein sogenanntes Item. Ein Item kann wiederum eine definierte oder eine undefinierte Länge haben. In einem Item sind ein- oder mehrere Presentation Data Values. Bei undefinierter Itemlänge wird bis zu einem Item Delimiter gelesen, welcher ebenfalls als Presentation Data Value mit Länge 0 dargestellt wird (Group-ID=0xFFFFE, Element-ID=0xE00D). In einer Sequence können mehrere Items vorkommen, es müssen aber nur diejenigen mit einem Item Delimiter versehen werden, welche eine undefinierte Länge haben. Eine Sequence mit undefinierter Länge wird durch einen Sequence Delimiter terminiert, dargestellt ebenfalls als Presentation Data Value mit Länge null (Group-ID=0xFFFFE, Element-ID=0xE0DD). Die Schwierigkeit bei dieser Codierung besteht darin, dass beim Verschicken einer Sequence die Gesamtlänge der Sequence trotzdem bekannt sein muss, da sie in die Nachrichtenlänge mit einfließt. Beispiel einer Sequence mit undefinierter Länge (Implicit Syntax, einfachheitshalber Big-Endian):

Byte	Field Name	Description
0-1	Group-D	Group-ID einer Sequence
2-3	Element-ID	Element-ID einer Sequence
4-7	PDV Length	Länge der PDV (0xFFFFFFFF)
8-9	Group-ID	Item Begin (0xFFFE)
10-11	Element-ID	Item Begin (0xE000)
12-15	Item Length	Länge der im Item enthaltenen PDV's (0xFFFFFFFF)
16-25	PDV	Zum Beispiel eine PDV mit einem Unsigned Short (2 Byte)
26-27	Group-ID	Item Delimiter (0xFFFE)
28-29	Element-ID	Item Delimiter (0xE00D)
30-33	PDV Length	Item Delimiter Length (0x00000000)
34-35	Group-ID	Sequence Delimiter (0xFFFE)
36-37	Element-ID	Sequence Delimiter (0xE0DD)
38-41	PDV Length	Sequence Delimiter Length (0x00000000)

Tabelle (6.17) Sequence Beispiel

ValueRepresentationSS ValueRepresentation für Signed Short. Vorzeichenbehafteter Short im 2er Komplement. Range von -2^{15} bis und mit $2^{15} - 1$. Die Länge ist fix zwei Bytes.

ValueRepresentationST ValueRepresentation für Short Text. Zeichenkette mit erlaubten Kontrollzeichen CR, LF, FF und ESC. «\» (Backslash) erlaubt. Maximal 1'024 Byte Länge.

ValueRepresentationTM ValueRepresentation für Time. Hat das Format HHMMSS.FFFFFFF (Stunde, Minute, Sekunde, «Fractional»). «Fractional» bis hin zu einer millionstel Sekunde. Zeichen von 0–9 und «.» (Punkt) erlaubt. Es ist erlaubt, Teile dieser ValueRepresentation wegzulassen (ausser HH). Dies bedeutet aber, dass alle Teile rechts davon auch weggelassen werden müssen. Maximale Länge beträgt 16 Bytes.

ValueRepresentationUI ValueRepresentation für Unique Identifier. Ein Unique Identifier ist eine normierte Zahl mit «.» (Punkt) getrennt (z. B. 1.2.10008.840.1.2). Darf Zahlen von 0–9 und «.» (Punkt) enthalten. Die maximale Länge beträgt 64 Bytes.

ValueRepresentationUL ValueRepresentation für Unsigned Long. Nicht-vorzeichenbehafteter Integer. Range von und mit 0 bis und mit $2^{32} - 1$. vier Byte fixe Länge.

ValueRepresentationUS ValueRepresentation für Unsigned Short. Nicht-vorzeichenbehafteter Short. Range von und mit 0 bis und mit $2^{16} - 1$. zwei Byte fixe Länge.

ValueRepresentationUT ValueRepresentation für Unlimited Text. Zeichenkette mit

erlaubten Kontrollzeichen CR, LF, FF und ESC. «\» (Backslash) erlaubt. Maximale Länge ist $2^{32} - 2$.

VR Factory Klasse für das Erstellen von ValueRepresentations anhand ihres «Value-Codes» (mittels Reflection).

SyntaxEncoder Basis Interface für das Encoding der Daten, abhängig vom Transfersyntax (6.5.5.5 auf Seite 92). Soll die Option offen halten, zu einem späteren Zeitpunkt andere Transfersyntaxen zu implementieren (momentan nur Implicit Little Endian).

ImplicitLittleEndianEncoder Encoder für die Transfersyntax Implicit Little Endian (siehe [Gla06]).

SyntaxDecoder Basis Interface für das Decoding der Daten, abhängig vom Transfersyntax (siehe 6.5.5.5 auf Seite 92). Soll die Option offen halten, zu einem späteren Zeitpunkt andere Transfersyntaxen zu implementieren (momentan nur Implicit Little Endian).

ImplicitLittleEndianDecoder Decoder für die Transfersyntax Implicit Little Endian (siehe [Gla06]).

6.5.7.6 Text Encoding

Der *DICOM*-Standard schreibt vor, dass Werte einer Werte-Repräsentation immer gerade sein müssen (gerade Länge). Deshalb müssen insbesondere Zeichenketten mit Leerzeichen (0x20) «gepadding» werden.

6.5.7.7 Transfersyntax Implicit Little Endian

Der Transfersyntax Implicit Little Endian stellt Zahlen im Little Endian Format dar ([Gla06] oder [Wik09a]). Dies beinhaltet die Darstellung aller Group- und Element-IDs, numerische Werte der Werte-Repräsentationen und die Längenangaben der Presentation Data Values. Die Längenangabe ist zusätzlich durch Implicit oder Explicit codiert (wie erklärt unter 6.5.5.5 auf Seite 92).

6.5.7.8 Abhängigkeiten von anderen Packages

dicom.util Spezielle Streams für *DICOM* spezifische Funktionalität.

6.5.8 Package Parser

6.5.8.1 Beschreibung des Package

Das Parser Package ist dafür verantwortlich, empfangene Daten von einem *DICOM*-Server zu übersetzen und die Daten in Objekte umzuwandeln. Dabei wird für das Parsen der

Association Nachrichten die gleiche Struktur aufgebaut wie die der Association Nachrichten selber. D.h. es gibt Nachrichten-Parser, welche Feld-Parser aufrufen, welche wiederum Unterfelder-Parser aufrufen können. Das Parsen der P-Data-TF (Composite-Find) Nachrichten wird über einen eigenen Nachrichten Parser geregelt, der den allgemeinen Teil liest und die Presentation Data Value Übersetzung an die Presentation Data Values delegiert, da diese das Wissen über ihre Darstellung beinhalten.

6.5.8.2 Diagramm

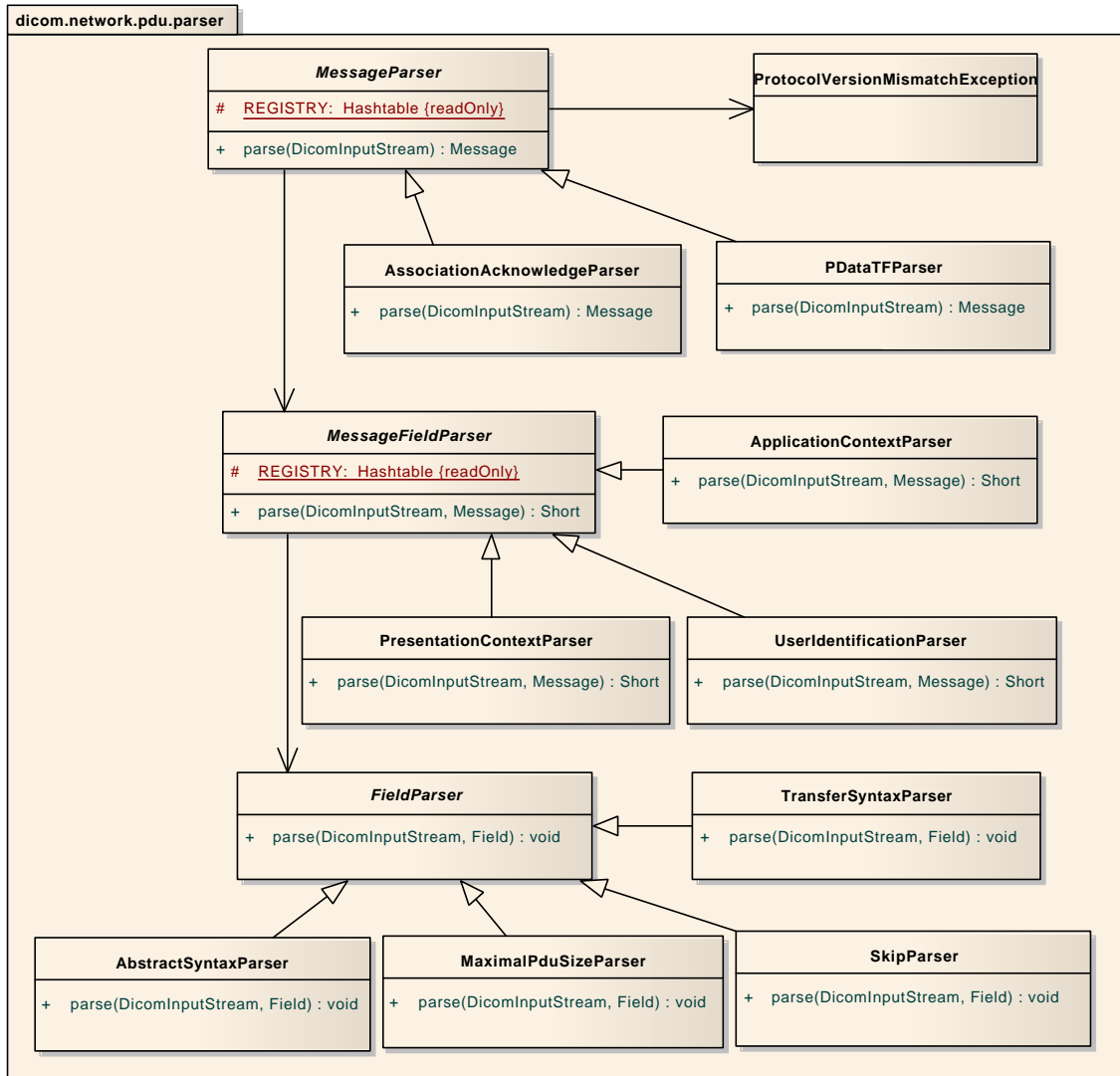


Abbildung (6.9) Klassendiagramm Package Parser

6.5.8.3 Schnittstellen

Der Einstieg in das Parser Package geschieht über die MessageFactory im network.pdu Package. Darin sind die Haupt-Nachrichtenparser registriert und können über die gelesene Nachrichten-ID angesprochen werden. Entweder ist dies eine AssociationAcknowledge Nachricht oder eine CFindMessage (P-Data-TF).

6.5.8.4 Klassen

MessageParser Abstrakte Basisklasse für Nachrichtenparser. Registriert alle Feld-Parser die von einem Nachrichten-Parser aufgerufen werden können. Zwingt Subklassen eine `parse` Methode zu implementieren, welche als Parameter einen `InputStream` nimmt, von dem die empfangenen Daten gelesen werden. Liefert als Ergebnis die geparte Nachricht.

MessageFieldParser Abstrakte Basisklasse für Feldparser. Ein Nachrichtenfeldparser parsed Felder, welche Unterfelder enthalten können. Demzufolge registriert die Klasse `MessageFieldParser` alle Feldparser für Unterfelder. Die Klasse zwingt Subklassen zur Implementierung einer `parse` Methode, welche als Parameter einen `InputStream` nimmt, von dem die Daten gelesen werden können und eine Nachricht, in welcher das erstellte Feld gesetzt werden kann. Als Rückgabewert wird die Anzahl der effektiv gelesenen Bytes zurück geliefert, um nur die Anzahl Bytes zu lesen, die auch wirklich der Nachrichtenlänge entsprechen.

FieldParser Abstrakte Basisklasse für Feldparser von Unterfeldern. Dies ist beispielsweise ein `TransferSyntaxField`, enthalten in einem `PresentationContextField`. Die Klasse zwingt Subklassen zur Implementierung einer `parse` Methode, welche als Parameter einen `InputStream` nimmt, von dem die Daten gelesen werden können und ein `Field`, in welchem das gelesene Feld gesetzt werden kann. Als Rückgabewert wird die Anzahl der effektiv gelesenen Bytes zurückgeliefert, um nur die Anzahl Bytes zu lesen, welche auch wirklich der Nachrichten bzw. Feldlänge entsprechen.

AbstractSyntaxParser Subklasse von `FieldParser`. Parst das `AbstractSyntaxField` einer `Association Response` und setzt dieses in einem `PresentationContextField`.

ApplicationContextParser Subklasse von `MessageFieldParser`. Parst das `ApplicationContextField` einer `Association Response` und setzt dieses in einer `AssociationMessage`.

AssociationAcknowledgeParser Subklasse von `MessageParser`. Parst eine `Association Acknowledge` Nachricht mit allen enthaltenen Feldern und Unterfeldern. Da die beiden Felder `ImplementationClassField` und `ClassUidField` für weitere Zwecke nicht benötigt werden, werden diese übersprungen (siehe Klasse `SkipParser`).

MaximalPduSizeParser Subklasse von `FieldParser`. Parst das `MaximalPduField` und setzt es in einem `UserInformationContextField`.

PDataTFParser Subklasse von `MessageParser`. Parst eine `Presentation Data Transfer` Nachricht mit allen `Presentation Data Values`. Konkret handelt es sich hier immer um eine `Composite-Find` Nachricht (`CFindMessage`).

PresentationContextParser Subklasse von MessageFieldParser. Parst das PresentationContextField einer Association Response und setzt dieses in einer Association-Message.

TransferSyntaxParser Subklasse von FieldParser. Parst das TransferSyntaxField und setzt es in einem PresentationContextField.

UserIdentificationParser Subklasse von MessageFieldParser. Parst das UserInformationContextField mit Unterfeldern und setzt es in einer Association Message.

SkipParser Subklasse von FieldParser. Liest lediglich die Länge eines Feldes und überspringt diese Bytes auf dem InputStream.

ProtocolVersionMismatchException Die ProtocolVersionMismatchException wird geworfen, wenn die vom Client eingesetzte Protokollversion nicht mit der vom Server übereinstimmt. Eine Kommunikation ist in diesem Falle nicht mehr möglich.

6.5.8.5 Abhängigkeiten von anderen Packages

dicom.network.pdu Nachrichten und Felder, welche geparkt werden.

dicom.util Spezielle Streams für *DICOM* spezifische Funktionalität.

6.5.9 Package Util (Allgemein)

6.5.9.1 Beschreibung des Package

Das Util Package beinhaltet einige Hilfsklassen, die regelmässig gebraucht werden. Es handelt sich dabei hauptsächlich um Funktionalität, die mit Java 1.6 mitgeliefert wird und hier in abgespeckter Version nachgebildet wurde.

6.5.9.2 Schnittstellen

Da es sich um ein utility Package handelt ist jede Klasse eine «Schnittstelle».

6.5.9.3 Subpackages

concurrent Enthält eine abgespeckte Version der Future-Klassen in Java 1.6.

6.5.9.4 Klassen

crypto.CipheredInputStream Entschlüsselt die Daten eines InputStreams, wobei für die Entschlüsselung ein BlockCipher (aus der Bouncycastle-Library, Kapitel 6.6.1 auf der nächsten Seite) übergeben werden muss.

crypto.CipheredOutputStream Verschlüsselt einen OutputStream mit Hilfe eine Block-Ciphers.

6.5.10 Package Util (DICOM)

6.5.10.1 Beschreibung des Package

Das Util Package von der *DICOM*-Funktionalität beinhaltet drei Hilfsklassen, welche als Hilfe für Serialisierung und Abspeicherung von ID / Werte Zuordnungen.

6.5.10.2 Schnittstellen

Da es sich um ein utility Package handelt ist jede Klasse eine «Schnittstelle».

6.5.10.3 Klassen

DicomInputStream Spezieller InputStream, der es ermöglicht eine fixe Anzahl von Bytes oder einen String fixer Länge zu lesen. Über ihn ist es auch möglich, *unsigned* Datentypen zu lesen.

DicomOutputStream Spezieller OutputStream, der es ermöglicht verschiedene *DICOM*-Spezifische Daten auf den Stream zu schreiben, um ihn danach in Byte-Form zu serialisieren (für die Netzwerkübertragung).

ShortPair Diese Klasse übernimmt die Schlüssel-Funktionalität für eine Hashtable, in der als Werte Werte-Repräsentationen stehen.

6.6 Externe Libraries

6.6.1 Bouncycastle

Ist genau genommen keine Library, da nur der Sourcecode ausgeliefert wird. Da es aber trotzdem wie eine Library benutzt wird, ist es hier trotzdem aufgeführt. Bouncycastle stellt eine grosse Auswahl an kryptographischen Funktionen zur Verfügung und wird für die Verschlüsselung der Reports verwendet. Es wurde Ausgewählt, da es ein speziell für Java *ME* optimierte Version davon gibt. Da die Source ausgeliefert werden, ist es auch möglich gezielt nur die Klassen einzubinden, die wirklich benötigt werden. Die Sourcen können unter folgender Adresse bezogen werden: http://www.bouncycastle.org/latest_releases.html

6.6.1.1 Verwendete Algorithmen

Damit die Daten sicher gespeichert sind wurden verschiedene Algorithmen eingesetzt:

SHA-256 Damit überprüft werden kann, ob das Passwort korrekt ist wird ein Hash davon gespeichert, der mit dem SHA-Algorithmus mit 256 Bit generiert wird.

AES Um den Report zu verschlüsseln wird der AES-Algorithmus mit einem 256 Bit langen Schlüssel verwendet. Der Schlüssel wird mit einem dafür geeigneten Generator generiert, der ebenfalls von Bouncycastle stammt.

RC4 Der Schlüssel wird mit dem RC4-Algorithmus verschlüsselt, wobei das Passwort zur Verschlüsselung verwendet wird. Die Sicherheit ist also schlussendlich von der Stärke des Passwortes abhängig.

6.7 Deployment

6.7.1 Files

Das Deployment der Applikation besteht aus der Installation der *Jar¹* Datei auf dem Endgerät. Da eine reine Client-Applikation entwickelt wurde, und die Serverseite nicht in dieser Arbeit vorhanden ist wird auf ein Deployment Diagramm verzichtet (Da nur auf mobilem Gerät installiert). Die Installation der Applikation geschieht über den Nokia Application Installer (eine Beschreibung dazu befindet sich im Handbuch des Gerätes).

6.8 Grössen und Leistung

6.8.1 Geschwindigkeit

Die Geschwindigkeit der Applikation ist nur bezüglich der übertragen im Netzwerk interessant, da die lokale Performance auch bei vielen Suchabfragen nicht beeinträchtigt wird (Ausser in Datensatzbereichen oberhalb der Anzahl 10000). Es wird deshalb auf das Kapitel 6.8.2 auf der nächsten Seite verwiesen.

6.8.1.1 GUI

Das *GUI* führt zeitintensive Operationen in einem separaten Thread aus, damit es nicht blockiert und immer sofort reagiert. Beim Aufbau einer Verbindung wird allerdings und *Java ME* das komplette System blockiert, inkl. aller Threads, weshalb das *GUI* während dieser nicht reagiert. Dies ist aber technisch nicht zu verhindern, da es keine asynchronen Varianten der Connection-Klasse gibt.

1 *Java Archive*

6.8.2 Netzwerk

Die Geschwindigkeit der Netzwerkübertragung ist abhängig von der W-LAN Kapazität und den darin aktiven Benutzern. Für eine Suchanfrage werden ohne das Übertragen der effektiv Gefundenen Daten (inklusive Anfrage) 6 Nachrichten übertragen:

- C->S: Association Request
- S->C: Association Acknowledge
- C->S: CFind Request Command
- C->S: CFind Request Data
- Server Data/Command Responses
- C->S: Association Release Request
- C->S: Association Release Acknowledge

C=Client, S=Server Die Grösse einer Gesamtübertragung kann also nicht komplett im voraus abgeschätzt werden, jedoch kann die Grösse in Abhängigkeit der Anzahl Gefundenen Datensätze sowie der anfangs mitgeschickten Transfersyntaxen ausgedrückt werden. Die fixen Komponenten (aus der vorherigen Auflistung) haben folgende Grössen (Für Unique Identifiers wird der Worst-Case fall von 64 Bytes verwendet):

- Association Request = 386 Byte + $((n - 1) * 68 \text{ Byte})$ für n =Anzahl Transfersyntaxen (da mindestens einer mitgeschickt werden muss, wird einer abgezogen).
- Association Response = 316 Byte + $(n * 68)$ für n =Anzahl Transfersyntaxen.
- CFind Request Command = 136 Byte
- CFind Request Data = 12 Byte + $(n * P_i)$ für n =Anzahl Presentation Data Values und P_i =Grösse der individuellen Presentation Data Value i , für $i=1 \dots n$.
- Server Responses = $(n + 1) * 136 \text{ Byte} + n * (u * P_i)$ für n =Anzahl gefundener Datensätze, u =Anzahl Presentation Data Values und P_i =Grösse der individuellen Presentation Data Value i , für $i=1 \dots n$.
- Association Release Request = 4 Byte
- Association Release Response = 4 Byte

Im schlechtesten Fall werden also $858 \text{ Byte} + (n * 68 \text{ Byte}) + (u * P_i) + n * (u * P_i) + (n + 1) * 136 \text{ Byte}$ übertragen. Die Basisabfrage ist kleiner als 1 Kilobyte und je nach Datenanzahl kommen für eine Query demzufolge etwa 1 - 9 KB (für 10000 Datensätze) zusammen. Dies ist weiter unter den vorhandenen W-LAN Kapazitäten heutiger Netze, und stellt kein Problem dar. Bei Abfragen mit vielen Treffern stellt hier also vorher der Server (für das Suchen) oder das Endgerät (für das Darstellen) einen Flaschenhals dar.

7 Testing

7.1 Dokument

7.1.1 Dokumentinformationen

Version: 1.0
Revision: Rev: 414
Status: Release
Erstellt am: 12.12.2009
Erstellt von: Mario Guhl

7.1.2 Dokumenthistory

Rev.	Datum	Wer	Änderung
0	12.12.2009	ythrier	Dokument erstellt.
363	12.12.2009	ythrier	Usability- und Performancetest
364	12.12.2009	ythrier	Systemtest
365	12.12.2009	ythrier	Testbeschreibung für RIS Emulator
365	17.12.2009	ythrier	Unittest Grafik

7.2 Einführung

7.2.1 Zweck

Dieses Dokument beinhaltet die durchgeführten Tests der Software Ihe4Mobile.

7.2.2 Gültigkeitsbereich

Dieses Dokument ist für die ganze Software gültig.

7.2.3 Übersicht

Nachfolgend wird auf die durchgeführten Tests für die Software Ihe4Mobile sowie auf die Codestatistiken eingegangen und eine Einführung in das Testen mit dem Validierungs-Toolkit gegeben.

7.3 Einleitung

Die in diesem Dokument durchgeführten Tests beziehen sich auf die Testplanung und Angaben aus dem Projektplan (siehe Kapitel 3.10.6 auf Seite 26).

7.3.1 Ergänzungen/Einschränkungen




Die unter Kapitel 3.10.6 auf Seite 26 aufgeführten Testszenarien und die zeitliche Durchführung mussten bedingt durch eingetretene Risiken angepasst werden. Es wird darauf in den entsprechenden Kapiteln der Testkategorien eingegangen. Für die Ursachen dieser Einschränkungen siehe Kapitel 3.6 auf Seite 10.





































7.4 Durchgeführte Tests

7.4.1 Unit Tests

ihe4mobile Code Coverage	Metrics:	CCN: 1.919	Methods: 714
		NCSS: 4.136	Classes: 116
		JVDC: 60.9%	Packages: 12

[Need help?](#)

	# Classes	Line Coverage	Branch Coverage	Method Coverage
All Packages	116	78.6% 	79.5% 	82.6% 

Package	# Classes	Line Coverage	Branch Coverage	Method Coverage
ch.hsr.ihe4mobile.business	2	96.7% 	100% 	96.2% 
ch.hsr.ihe4mobile.business.entities	14	69.1% 	68.1% 	77.4% 
ch.hsr.ihe4mobile.business.services	3	96.3% 	100% 	90.9% 
ch.hsr.ihe4mobile.dicom.network	4	82% 	92.3% 	83.3% 
ch.hsr.ihe4mobile.dicom.network.pdu	27	94.4% 	100% 	92.9% 
ch.hsr.ihe4mobile.dicom.network.pdu.parser	13	99.1% 	100% 	100% 
ch.hsr.ihe4mobile.dicom.network.vr	28	85.6% 	82.1% 	93.7% 
ch.hsr.ihe4mobile.dicom.util	4	100% 	100% 	100% 
ch.hsr.ihe4mobile.persistence.storage	6	48.4% 	100% 	37.2% 
ch.hsr.ihe4mobile.util	5	87.6% 	100% 	95.2% 
ch.hsr.ihe4mobile.util.concurrent	8	0% 	0% 	0% 
ch.hsr.ihe4mobile.util.crypto	2	88.6% 	100% 	85.7% 

Legend:

CCN: Cyclomatic Complexity

NCSS: Non-Commenting Source Statement

JVDC: JaVaDoc Comment

Reports generated by [Cobertura for J2ME](#) version 1.1.0RC2.
17.12.09 19:58

Abbildung (7.1) Testabdeckung

Die Testabdeckung ist gesamthaft in einem vertretbaren Rahmen. Die Packages «Storage» und «Concurrent» konnten leider kaum oder nicht getestet werden, da bei Storage für das Testen ein Adapter erstellt werden musste, um ein Mock-Objekt zu kreieren. Dies hat somit

das Ergebnis verfälscht da der Adapter 0% Testabdeckung hat. Um die Asynchronität im Package «Concurrent» zu testen, waren Unittests nicht geeignet.

7.4.2 Usability Tests

Da erst am Ende der letzten Construction-Iteration eine wirklich testbare Version der Software vorhanden war, wurde die Anzahl der Ursprünglich geplanten Usability-Tests von drei auf eins reduziert. Ausserdem konnte nur noch eine externe Testpersonen für den Test herangezogen werden.

7.4.2.1 Fragebogen

Der Erfüllungsgrad bezieht sich hier auf folgende Punkte:

- Ist die Funktion auffindbar?
- Ist sie verständlich?
- Ist die Navigierbarkeit intuitiv?

Nr.	Frage / Auftrag	Kommentar	Erfüllungsgrad ^a
1	Eingabe der Konfiguration	Geben Sie Konfigurationsdaten an (vom Betreuer erhalten)	-
2	Worklist-Suche	Suchen Sie alle Arbeitsschritte	-
3	Worklist-Suche	Suchen Sie anhand von Vorname	-
4	Worklist-Suche	Suchen Sie anhand von Nachname	-
5	Worklist-Suche	Suchen Sie anhand von Vor- und Nachname	-
6	Worklist-Suche	Suchen Sie anhand von einer Modalität	-
7	Taskdetails	Lassen Sie sich die Taskdetails anzeigen	-
8	Taskdetails	Fügen Sie einen Report hinzu	-
9	Taskdetails	Lassen Sie sich die Patientendetails anzeigen	-
10	Taskdetails	Markieren Sie den Task als erledigt	-
11	Navigation	Navigieren Sie durch die gesamte Applikation	-

^a 0%–100%

Tabelle (7.1) Fragebogen Usability Tests

Nr.	Frage / Auftrag	Kommentar	Erfüllungsgrad
1	Eingabe der Konfiguration	Geben Sie Konfigurationsdaten an (vom Betreuer erhalten)	95 %
2	Worklist-Suche	Suchen Sie alle Arbeitsschritte	60 %
3	Worklist-Suche	Suchen Sie anhand von Vorname	90 %
4	Worklist-Suche	Suchen Sie anhand von Nachname	30 %
5	Worklist-Suche	Suchen Sie anhand von Vor- und Nachname	30 %
6	Worklist-Suche	Suchen Sie anhand von einer Modalität	80 %
7	Taskdetails	Lassen Sie sich die Taskdetails anzeigen	100 %
8	Taskdetails	Fügen Sie einen Report hinzu	100 %
9	Taskdetails	Lassen Sie sich die Patientendetails anzeigen	100 %
10	Taskdetails	Markieren Sie den Task als erledigt	100 %
11	Navigation	Navigieren Sie durch die gesamte Applikation	90 %

Tabelle (7.2) Fragebogen Usability Tests Corsin Camichel

Kommentar

- In der Konfiguration sollte Server anstelle von Host stehen
- Bei «Taskliste» denkt man nicht an eine Suche. Man sollte dies anders benennen
- Der «Exit» anstelle von «Back» Button stört enorm (bei Taskliste)

Anmerkung: Leider ist das Umstellen der Buttons auf dem Nokia N97 nicht möglich, deshalb ist der Exit Knopf nach wie vor an der selben stelle.

7.4.3 Performance Tests

Da erst am Ende der letzten Construction-Iteration eine wirklich testbare Version der Software vorhanden war, wurde die Anzahl der Ursprünglich geplanten Performance-Tests von zwei auf eins reduziert. Die Werte wurden anhand des Java™ ME Profilers ausgelesen (Eclipse Console Output).

Nr.	Test	Wert	Erfüllungsgrad
1	Hauptspeicher	3 MB	100 %
2	Festplatte (Jar Grösse)	0.3 MB	100 %

Tabelle (7.3) Harte Grenzen Performance Test

7.4.4 System Tests

7.4.4.1 Use Case Tests

Dieser Testabschnitt der System Tests bezieht sich auf die Use Case Funktionalität (siehe Kapitel 4.9 auf Seite 41).

<i>UC</i>	<i>Titel</i>	<i>Erfüllungsgrad</i>
1	Setting up a connection to a <i>DICOM</i> -Server	100 %
2	View tasks for all application entities	100 %
3	Proceed a task	75 % (kein Abschliessen auf Server)
4	Search for patient	Optional
5	View patient data	100 % (möglich durch Task-Suche)
6	View patient study	optional

Tabelle (7.4) Use Case Tests

7.4.4.2 Anforderungstest

Der Anforderungstest bezieht sich auf die Überprüfungen der Muss-Anforderungen (erfüllt/nicht erfüllt).

Muss Anforderungen

<i>Anforderung</i>	<i>Erfüllungsgrad</i>
Anmelden an einen zentralen <i>DICOM</i> -Server	100 %
Abfragen von Behandlungsaufträgen für verschiedene Untersuchungsstationen	100 %
Bestätigen von durchgeführten Behandlungsaufträgen	50 % (nur lokal)
Integration in bestehende Lösung	0 % (es wurde nur mit dem Validierungstoolkit gearbeitet)

Tabelle (7.5) Anforderungstests – Muss

7.5 Testauswertung

7.5.1 Erfüllungsgrad

<i>Kategorie</i>	<i>Erreichtes Resultat</i>
Unit Tests (Coverage)	78 %
Usability Tests	71 %
Performance Tests	100 %
System Tests	80 %
Durchschnitt	82 %

Tabelle (7.6) Erfüllungsgrad

7.5.2 Bewertung

Die angestrebten 95 % konnten leider nicht erreicht werden, da es uns nicht als Sinnvoll erschien im Anbetracht des benötigten Aufwandes. Um bei den Unittests ein höheres Resultat zu erreichen, hätten beispielsweise auch unsinnigerweise Get-/Set-Methoden getestet werden müssen. Es gab auch Komponenten, die nahezu «untestbar» waren. Dies hat das reale Testergebnis verfälscht. Beim Usability-Test wurden die niedrigsten Werte durch die ungeschickte Platzierung von *GUI*-Komponenten verursacht, auf welche wir aber keinen Einfluss nehmen konnten, da dies vom System abhängig ist.

7.6 Codestatistiken

In diesem Abschnitt werden einige Kenngrößen zum Sourcecode der Software aufgeführt:

<i>Was</i>	<i>Wert</i>
Packages	21 (7 Externer Code)
Classes	163 (26 Externer Code)
<i>NCSS</i> ¹	~5300
Revisions	~400

Tabelle (7.7) Codestatistiken

7.7 Testen mit *RIS* Emulator

Die Applikation wurde mit Hilfe des *RIS*¹ Emulators von DVTk [dvt]) getestet. Auf dieser Website sind noch andere hilfreiche Tools vorhanden, auf die nicht weiter eingegangen wird. Nachfolgend ist ein Testaufbau beschrieben, der es ermöglicht, ohne das Aufsetzen eines konkreten *DICOM*-Informationssystems, zu sehen, wie die Applikation funktioniert und auch zu testen:

- Downloaden und installieren des *RIS* Emulators (Standardinstallation)
- Kopieren der Testdateien ([GT09]) nach `<installation directory>\ data\ worklist`
- Starten des *RIS* Emulators und Konfigurieren des Worklist local Port (muss erreichbar sein über das W-LAN, möglicherweise muss auch die Firewall konfiguriert werden)
- Auf dem Smartphone kann nun die Adresse des Rechners und der eingetragene Port konfiguriert werden und dann zum Emulator verbunden werden

1 *Radiologieinformationssystem*

8 Technischer Bericht

8.1 Dokument

8.1.1 Dokumentinformationen

Version: 1.0
Revision: Rev: 416
Status: Release
Erstellt am: 08.12.2009
Erstellt von: Mario Guhl

8.1.2 Dokumenthistory

Rev.	Datum	Wer	Änderung
0	08.12.2009	ythrier	Dokument erstellt.
354	09.12.2009	ythrier	Erste Version fertiggestellt.
355	10.12.2009	ythrier	Überarbeitet.
360	14.12.2009	ythrier	Nach Feedback von A.D. überarbeitet und Rechtschreibung
386	15.12.2009	ythrier	Projektmanagement und Zeitauswertung ergänzt

8.2 Einführung

8.2.1 Zweck

Dieses Dokument ist der technische Bericht der Software Ihe4Mobile.

8.2.2 Gültigkeitsbereich

Die in diesem Dokument gemachten Angaben gelten für das gesamte Projekt.

8.2.3 Übersicht

In diesem Dokument wird auf die Studienarbeit Informatik «Workflowsteuerung und Bilddarstellung für medizinische Informationssysteme mit mobilen Geräten» bezüglich der Aufgaben-/Problemstellung, der gemachten Vorarbeiten und auf die Ergebnisse eingegangen. Zuletzt ist eine Schlussfolgerung mit Bewertung sowie Rückblick/Ausblick vorhanden.

8.3 Einführung in die Arbeit

8.3.1 Problem-/Aufgabenstellung

8.3.1.1 Ausgeschriebene Studienarbeit

Die Studienarbeit hat das Ziel, die im Medizinbereich einzughaltende Vernetzung und den Gebrauch von Informationssystemen auf einer «mobilen» Plattform verfügbar zu machen. Dabei soll mittels des *DICOM*-Standards (siehe [Nat]) auf einem medizinischen Informationssystem Arbeitsschritte für einen Patienten an einer *Modalität* gesucht und auf der mobilen Plattform dargestellt werden können.

8.3.1.2 Konkrete Aufgabenstellung

Die konkrete Aufgabenstellung wurde nach dem Kick-Off Meeting auf folgende Muss-Kriterien festgelegt:

- Anmeldung an einen *DICOM*-Server
- Abfragen von Behandlungsaufträgen («planned procedure steps») für verschiedene Untersuchungsstationen («application entities»)
- Bestätigung von durchgeführten Behandlungsaufträgen («performed procedure steps»), welche lediglich textuelle Resultate liefern
- Integration in bestehende Lösung [Vet09]

Die optionalen Anforderungen:

- Suchen/Abfragen von Patienteninformationen
- Abfragen von durchgeführten Untersuchungen/Behandlungen («performed procedure steps») ohne Anzeige von Bildern
- Abfragen von durchgeführten Behandlungsaufträgen mit Anzeige von Bilddateien
- Lokales Cachen der Daten, damit die Erfassungsaufgaben auch ohne eine Serververbindung möglich sind
- Login-/Authentifizierungsprozedur auf Clientseite um Datenmissbrauch auf lokaler Basis zu verhindern (mit Option auf zentralisierte Rechteverwaltung)
- Verschlüsselung von lokal gespeicherten Daten
- Update Mechanismus und zentrale Konfigurationsverwaltung

8.3.1.3 Vorarbeiten

Vorarbeiten in diesem Bereich wurden von Hr. Florian Vetter durchgeführt ([Vet09]). Hr. Vetter hat bezüglich der Serverseite bereits eine Implementierung bzw. Konfiguration eines *DICOM*-Servers vorgenommen, auf den in dieser Arbeit zugegriffen werden sollte. Ebenfalls hat er eine kleine *DICOM* einföhrung geschrieben, welche einen schmalen Einblick in die ersten Begrifflichkeiten bietet.

8.4 Lösungsskizze

Die Technologie beruht auf Java™ Mobile Edition mit *CLDC*¹ 1.1 und *MIDP*² 2.1. Es wurde ein Nokia N97 mit Symbian S60 verwendet. Der architektonische Ansatz besteht aus einem simple 3-Layer Modell, das jedoch in diesen einzelnen Ebenen stark unterteilt ist. Grundsätzlich gibt es GUI, Business und Storage. Der GUI-Layer ist ein einzelnes Package, das lediglich der Visualisierung dient. Business ist in die Dienste die zur Verfügung gestellt werden, Programmeigenschaften und die Abbildung der *DICOM*-Entitäten unterteilt. Im Storage-Layer geschieht eine Unterteilung zwischen Persistierung und Netzwerkverhalten. Somit ist in Network die gesamte *DICOM* Kommunikationsfunktionalität vorhanden. Es wird hier generell zwischen den Darstellungen von Werte und dem Aufbau der Kommunikationsnachrichten unterschieden. Besonders erwähnenswert ist hier einerseits die Konfigurierung der Objekt / Wert Codierung anhand eines Konfigurationsfile, andererseits auch die Überprüfung der Werte-Representationen anhand der im Standard vorgeschriebenen Regeln. Diese beiden Konzepte werden anhand von einer Reflection gesteuerten Factory Klasse verbunden, welche es ermöglicht ad-hoc die richtigen Zuordnungen für die im System benötigten Daten zu erstellen.

8.5 Vorgehensweise

Die ursprüngliche Planung war im Vergleich zum Effektiv geleisteten bis hin zur mitte der ersten Elaboration-Phase sehr ähnlich. Wir haben dann erste Tests bezüglich der *DICOM*-Library dcm4che2 und Java™ Mobile Edition durchgeführt. Es hat sich schnell heraus gestellt, dass diese mit den einschränkungen von Java™ Mobile Edition (Java 1.4) nicht lauffähig ist. Eine direkte Portierung kam nach einer Analyse des Sourcecodes nicht in Frage. Der Entschluss die benötigte *DICOM*-Funktionalität selber zu implementieren hat trotz der früh herbei geföhrten Entscheidung zu sehr grossen Abweichungen vom Zeitplan geföhrt. Der Hauptteil der Arbeit bestand nun darin funktionalität zu implementieren, welche ursprünglich übernommen werden sollte. Daraus resultierend musste die Planung insofern

1 *Connected Limited Device Configuration*

2 *Mobile Information Device Profile*

angepasst werden, dass mehr Zeit für das Technologiestudium sowie Netzwerkfunktionalität verwendet werden musste. Auch bei der Infrastruktur war die Planung und das Ist stark abgewichen. Das Erstellen der L^AT_EX Vorlagen, sowie das Aufsetzen und Konfigurieren des Buildsystems nahmen bedeutend mehr Zeit in Anspruch, als ursprünglich geplant. Grundsätzlich hätte die Planung genauer getätigt werden sollen, bzw. in die Analyse-Phasen hätte mehr Zeit investiert werden sollen, um spätere Engpässe zu vermeiden. In Summe jedoch, konnte das Projekt zu einem zufriedenstellendem Ende gebracht werden.

8.6 Ergebnisse

8.6.1 Effektives Resultat

Das Endresultat ist ein auf JavaTM Mobile Edition lauffähiger Client, welcher die Fähigkeit hat, auf einem *DICOM*-Server anhand des Namens einer *Modalität* und eines Patienten Arbeitsschritte zu suchen und diese anzuzeigen. Die Adresse und der Port des Servers kann frei konfiguriert werden. Zusätzlich kann ein Passwort für das Verschlüsseln der lokalen Daten angegeben werden. Die Daten, welche lokal abgespeichert sind (Reports), werden anhand einer Autosave-Funktion automatisch regelmässig gespeichert. Die Applikation besitzt zwei Konfigurationsdateien:

InformationObjectDefinition: Anhand dieser Datei können selbständig zusätzliche Werte-Attribute mit Werte-Representation konfiguriert werden (Information Object Definition: Kapitel 6.5.6.4 auf Seite 96, Value Representation: Kapitel 6.5.7 auf Seite 105).

ModalityDefinitions: Anhand dieser Datei können die *Modalitäten*, welche in der Suchmaske für Arbeitsschritte zur Auswahl stehen konfiguriert werden.

Vom *DICOM*-Standard wurden die Assoziierung (A-Association) ohne extended Negotiation implementiert. Standardmässig wird als Assoziierungskontext ein «Modality Worklist Find» angegeben, Code-Seitig könnte man dies jedoch mit jedem beliebigen Identifikator ersetzen. Da das Abbrechen von I/O-Operation mittels JavaTM Mobile Edition nicht direkt möglich war, wurde auf das Implementieren der Abbruchnachrichten verzichtet (A-Abort, -Reject und -Release sind vorhanden). Darauf aufbauend wurde die Composite-Finde Methodik implementiert. Dies beinhaltet das Aufbauen und Versenden einer Command und Data P-Data-TF Nachricht mit angehängten Presentation Data Values (Kapitel 6.5.6.3 auf Seite 96) für Such- und Matching Attribute.

8.6.2 Erfüllte Anforderungen

Von den geplanten Muss-Kriterien konnten nicht alle Punkte umgesetzt werden (Siehe 8.6.3 auf der nächsten Seite). Folgende Muss-Punkte wurden umgesetzt:

- Anmeldung an einen *DICOM*-Server (A-Association).
- Abfragen von Behandlungsaufträgen («planned procedure steps») für verschiedene Untersuchungsstationen («application entities»). Die Suche kann hierbei anhand der Modalität und des Patientennamen geschehen (Composite-Find).
- Bestätigung von durchgeführten Behandlungsaufträgen («performed procedure steps»), welche lediglich textuelle Resultate liefern. Die Bestätigung der Behandlungsaufträgen ist nur «logischer» Natur. d. h., die Arbeitsschritte werden nur lokal als beendet markiert. Eine Übertragung dieses Statuses auf den Server wurde nicht implementiert.

Von den optionalen Anforderungen wurde umgesetzt:

- Lokales Cachen der Daten, damit die Erfassungsaufgaben auch ohne eine Serververbindung möglich sind. Es wurde ein lokales Cachen der Reports für Arbeitsschritte eingeführt. Da das Abschliessen von Arbeitsschritten nicht an den Server übertragen wird, ist das lokale Cachen dieser nicht verwirklicht.
- Login-/Authorisierungsprozedur auf Clientseite um Datenmissbrauch auf lokaler Basis zu verhindern.
- Verschlüsselung von lokal gespeicherten Daten

8.6.3 Offene Punkte/Probleme

Ursprünglich geplant war, für die *DICOM* Kommunikationsfunktionalität die Open-Source Library *dcm4che2* zu verwenden (siehe [Zei]). Da diese jedoch nicht mit Java™ Mobile Edition lauffähig war, und auch keine andere Java library die entsprechende Funktionalität portabilerbar anbotete, wurde die benötigte Funktionalität selber implementiert. Dies hatte ein intensiveres Einarbeiten in die *DICOM*-Funktionalität als ursprünglich geplant zur Folge. Der *DICOM*-Standard ist als offener Industriestandard über die Jahre hinweg angewachsen, und hat viele schwer umsetzbare Punkte/Sektionen. Auch historisch bedingte, unseres erachtens nach «Altlasten», welche jedoch nach wie vor Teil des Standards und unabdingbar für Kernfunktionalitäten sind, steigern die Komplexität enorm. Dies führte dazu, dass der Hauptaufwand für die Implementierung der Kommunikation mit dem Server aufgewendet werden musste und dadurch die Umsetzung aller Anforderungspunkte im vorgegebenen Zeitrahmen nicht möglich war. Resultierend daraus sind folgende Punkte nicht umgesetzt worden:

- Bestätigung von durchgeführten Behandlungsaufträgen («performed procedure steps») auf dem Server
- Integration in bestehende Lösung ([Vet09])

- Suchen/Abfragen von Patienteninformationen
- Abfragen von durchgeführten Untersuchungen/Behandlungen («performed procedure steps») ohne Anzeige von Bildern
- Abfragen von durchgeführten Behandlungsaufträgen mit Anzeige von Bilddateien
- Update Mechanismus und zentrale Konfigurationsverwaltung

8.6.4 Vergleich Soll-/Ist-Resultat

Vergleich des Soll-/Ist-Resultats bezüglich nicht erfüllter Punkte und deren resultierende Einschränkung.

<i>Geplantes Ergebnis</i>	<i>Erreichtes Ergebnis</i>	<i>Resultierende Einschränkung</i>
Worklist Task Abfrage und Abschluss	Abfrage und lokales abschliessen	Möglicherweise gleicher Schritt zwei Mal in der Liste, sowie kein Übertragen von Ergebnissen auf den Server.
Integration in bestehende Lösung	Validierung gegen <i>DVTk</i>	Nur an «fiktivem» Endpunkt überprüft. Möglicherweise nicht mit jeder <i>DICOM</i> Serverlösung kompatibel.
«performed procedure steps» ohne/mit Bilder	Nur nicht durchgeführte	Keine Möglichkeit «alte» Schritte mit der Applikation anzuschauen.
Zentraler Update Mechanismus	-	Neue Versionen der Software müssen Manuel auf den Endgeräten aufgespielt werden.

Tabelle (8.1) Vergleich Soll-/Ist-Resultat

8.7 Schlussfolgerungen

8.7.1 Bewertung des Projekts

Trotz der Komplikationen sehen wir die ganze Arbeit als Erfolg. Es ist durchaus praktikabel ein Worklist-Management auf einem Smartphone oder anderen mobilen Geräten bei sich zu haben, um dadurch flexibel vor Ort und Stelle direkt auf Daten zuzugreifen (Papierlos). Da die Geräte in diesem Bereich auch immer leistungsfähiger werden, steht auch komplexeren Applikationen nichts im Wege - lediglich dass generelle Verwalten des Informationssystems wird nicht in absehbarer Zeit die Domäne der normalen Desktop Computer durchbrechen. Ins Gewicht fällt jedoch auch die Erkenntnis, dass sensible Anwendungen oder kommerziell wichtige Applikationen - zumindest bei Smartphones - entwicklungstechnisch schwer zu realisieren sind. Notwendige Grenzen der Umsetzungssprachen durch Speicher- und

Energieoptimierung (Akkulaufzeit) schränken die gewohnte Arbeit ein und führen zu Mehraufwand auf Implementationsebene. Die Applikation Ihe4Mobile ist sicher ein guter Schritt in die richtige Richtung, wird sich aber aufgrund des Mangels gewisser Funktionalitäten nicht durchsetzen können. Sie zeigt jedoch, dass die Möglichkeiten in diesem Bereich sehr gross sind. In dem Sinne, ist das Produkt eher als «Beweis des Nutzens» zu sehen und nicht als eine «verkaufbare» Software.

8.7.2 Bewertung Methodik und Tools

RUP und *TDD*¹ haben sich über die Projektlaufzeit nur teilweise bewährt. Insbesondere *TDD* hat bestimmte «Hemmschwellen» die zu überwinden sind. Eine Testabdeckung von 100% ist zwar durchaus wünschenswert, aber Aufwand/Nutzen muss in jedem Fall analysiert werden und dann entschieden welcher Grad erreicht werden will. Dies kann teilweise nicht bei der Planung sondern erst bei der Realisierung festgelegt werden. *RUP* ist schreibtechnisch ein zu grosser Aufwand für das was man kriegt; Für Folgeprojekte empfehlen sich in dem Sinne vermutlich andere Methodiken (Extreme Programming, Scrum). Die angewendeten Tools haben sich grösstenteils bewährt und die Arbeit enorm erleichtert. Insbesondere \LaTeX und das Buildsystem waren hilfreich für schnelles und effizientes Arbeiten. Durch die Benutzung von Bugzilla als Bugtracker war auch das Verfolgen von Fehlern sehr angenehm und man konnte diese Stück für Stück abarbeiten. Code-Coverage war nützlich als Feedback und konnte auch sehr gut in den Arbeitsprozess integriert werden. Die restlichen Tools (Java Code Style Check, Copy Paste Detector und Codestriker) waren «nice to have», aber wurden mehr am Rande gelegentlich überprüft. Codestriker hat sehr viel Potential und wenn regelmässig angewendet einen sehr positiven Effekt auf das Projekt; Dies muss dann aber auch konsequent durchgeführt werden.

8.7.3 Zeitauswertung

Nachfolgend befinden sich die Zeitauswertungen bezüglich des Projekts. Dies beinhaltet den Gesamtvergleich von Plan-/Ist-Stunden und auch aufgeteilt auf die Arbeitspakete.

¹ *Test Driven Development*

8.7.3.1 Zeitauswertung Plan/Ist

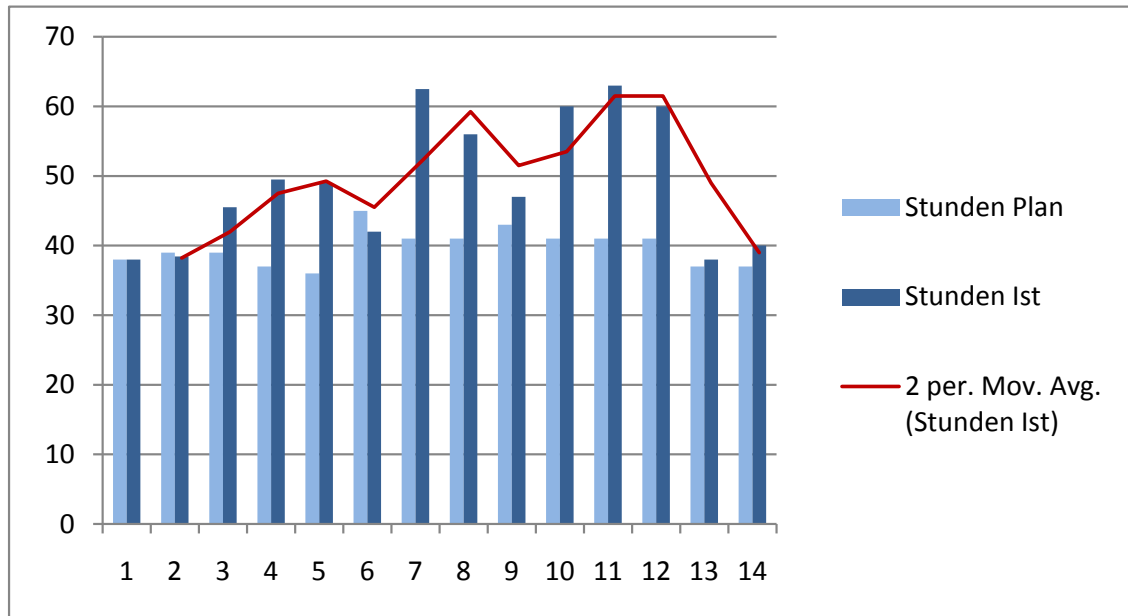


Abbildung (8.1) Plan/Ist Vergleich der Stunden pro Woche

Interpretation

Zu Beginn zeichnete sich eine gute Übereinstimmung zwischen Plan und Ist. Ab der dritten Woche kamen die Mehraufwände bezüglich L^AT_EX Vorlagen und Buildsystem (Infrastruktur allgemein) hinzu. Ende Woche 4 und fortlaufend kam die Problematik der DICOM-Library auf und dies zog sich bis Ende Woche 12 (ende Construction 3) durch. Die Transition-Phase entsprach dann wieder dem geplanten Ablauf. Betrachtet man sich die Trendlinie, sieht man dass die Durchschnittliche Zeit pro Woche ansteigt, sich jedoch gegen Ende wieder in den geplanten Rahmen einpendelt.

8.7.3.2 Zeitauswertung Arbeitspakete

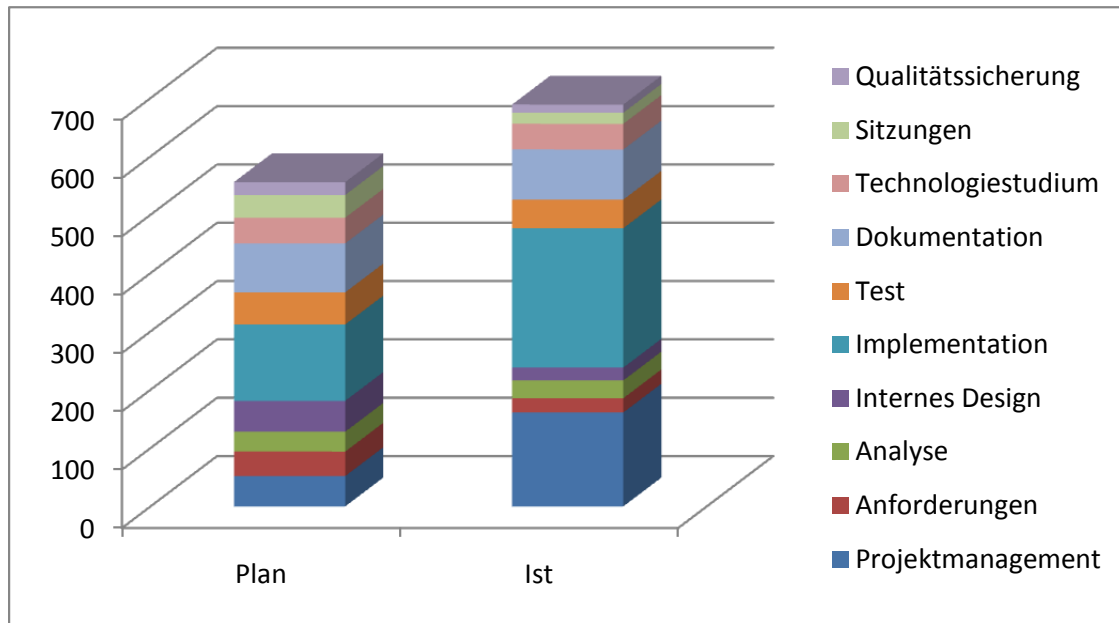


Abbildung (8.2) Plan/Ist Vergleich der Stunden pro Arbeitspaket

Interpretation

Die Verhältnisse der einzelnen Arbeitspakete im Vergleich Plan/Ist korrelieren bei den meisten Kategorien. Im Projektmanagement sind bedeutend mehr Stunden aufgewendet worden, durch den Mehraufwand in der Infrastruktur. Gelitten hat die Qualitätssicherung, die leider sehr niedrig ist. Für uns ist klar, dass für weitere Projekte diese besser in den Arbeitsprozess integriert werden muss.

8.7.4 Lessons Learned

Als lerntechnischen «Profit» aus dem Projekt können wir folgende Punkte aufführen:

- Arbeiten mit \LaTeX (inkl. Erstellen von Vorlagen)
- Umgang mit Software-Entwicklung auf mobilen Plattformen
- Benutzen und Einrichten eines Buildsystem für Continuous Integration
- Einarbeiten in (Industrie-)Standard
- Erstellen von Automations-Skripten für \LaTeX und generieren von Metrik/anderen Auswertungen

8.7.5 Rückblick

Rückblickend betrachtet war der Aufwand im Vergleich zum Nutzen nicht optimal. Es musste viel Zeit investiert werden um die Kommunikation des *DICOM*-Protokolls zu verstehen. Eine längere Analyse hätte zu einem besseren Resultat geführt, aber dies war zeitlich gesehen nicht möglich, da die Projektdauer fix auf 14 Wochen beschränkt ist. Die Benutzung von Java™ Mobile Edition hat auch seine Tücken, beispielsweise beim Unit-Testing, da hier alle Tests über ein Switch-Case Statement aufgerufen, und im Konstruktor der Basisklasse die Anzahl dieser übergeben werden müssen. In Summe war der Entwicklungsprozess an manchen Stellen sehr «lästig» und Nerven aufreibend, hat aber auch viel Spass und Erfolgserlebnisse gebracht.

8.7.6 Ausblick

Es ist durchaus denkbar, Zeit in die Weiterentwicklung zu investieren, und zusätzliche Funktionalitäten wie das nicht implementierte Abschliessen von Arbeitsschritten auf dem Server, oder sogar abrufen/speichern von Bilddaten hinzuzufügen. Für bestimmte Medizinbereiche könnte dies einen bedeutenden Mehrnutzen bringen; Denkbar wäre hier eine Anwendung in der Dermatologie, in der mit der integrierten Gerätekamera, die heutzutage in nahezu jedem Smartphone vorhanden ist, Fotos von Hautstellen erstellt, und dann im Informationssystem abgelegt werden. Diese würde jedoch die Implementierung weiterer *DICOM* Funktionalitäten bedeuten, die momentan nicht in der Applikation vorhanden sind. Da diese nicht im Kontext der Arbeit lagen, wurde auch nicht überprüft, inwiefern die Anpassbarkeit auf diese Dienste gewährleistet ist.

9 Persönliche Berichte

9.1 Dokument

9.1.1 Dokumentinformationen

Version: 1.0
Revision: Rev: 412
Status: Release
Erstellt am: 13.12.2009
Erstellt von: Mario Guhl

9.1.2 Dokumenthistory

Rev.	Datum	Wer	Änderung
0	13.12.2009	ythrier	Dokument erstellt.
368	13.12.2009	ythrier	Vorlage für M. Guhl und Y. Thrier erstellt. Eigenen Report verfasst
369	17.12.2009	mguhl	Persönlicher Bericht erstellt

9.2 Persönlicher Bericht: Mario Guhl

9.2.1 Einleitung

Diese Studienarbeit habe ich ausgewählt, da ich bis jetzt noch nie etwas für ein Embedded Device entwickelt habe und somit möglichst viel davon profitieren wollte.

9.2.2 Die Studienarbeit

Am Anfang der Studienarbeit waren wir voller Ideen und mussten uns sehr zurücknehmen um bei den Anforderungen nicht über die Stränge zu schlagen, was sich wie sich später herausstellte eine gute Idee war. Da wir als Vorgabe für die Studienarbeit einen Buildserver betreiben sollten und die \LaTeX Vorlagen auch einiges an Zeit beanspruchten war ich die ersten vier Wochen nebst dem erstellen der Dokumente vor allem damit beschäftigt. In dieser Zeit machten sich bereits die ersten Probleme bemerkbar, da die Einrichtung des Buildservers wesentlich komplizierter war als zunächst vermutet. Dies lag einerseits daran, dass die Plugins, die auf der Homepage von CruiseControl keine richtigen Plugins waren und die Zusammenarbeit mit dem Buildserver selber mittels Ant-Script umgesetzt werden musste. Andererseits wirkte sich die abgespeckte Funktionalität auch auf das Buildscript aus, da Code-Coverage nicht wie bei Java 1.6 in die eclipse-Umgebung integriert werden konnte, sondern ebenfalls nur mittels Buildscript umgesetzt werden konnte. Dank vieler

Überstunden konnte dann aber auch dies umgesetzt werden. Als wir dann mit der Umsetzung des Prototypen beginnen konnten merkten wir, dass die Library, welches den *DICOM*-Standard implementiert, nicht auf Java *ME* eingesetzt werden konnte. Dies änderte unsere Pläne völlig, denn unsere Hauptaufgabe bestand nun nicht mehr darin einen Worklist-Service zu entwickeln, sondern den *DICOM*-Standard bzw. Teile davon zu implementieren, damit wir die eigentliche Aufgabe überhaupt lösen konnten. Dies führte natürlich dazu, dass die ursprünglichen Anforderungen erheblich gekürzt werden mussten. Da die Implementation eines Standards vor allem eine Fleissarbeit ist beeinträchtigte dies dann auch die Motivation, da wir so keine Zeit hatten die Unterschiede zu einem PC-System richtig kennenzulernen, was sehr schade war.

9.2.3 Rückblick

Rückblickend würde ich sage, dass ich zu viel Zeit in die Aufsetzung des Buildservers eingesetzt hatte. Ich weiss aber nicht, ob ich es nächstes Mal anders machen würde, da ich bei Herausforderungen nicht gerne Aufgabe und der Buildserver stellte für mich bei dieser Studienarbeit die grösste Herausforderung dar. Für die \LaTeX -Vorlagen habe ich gegenüber dem Projektplan ebenfalls viel zu viel Zeit investiert, was ich aber nicht bereue, da wir diese ja für die Bachelorarbeit weiter Verwenden können. Zudem bin ich mir überhaupt nicht sicher ob es bei einem Dokument dieser Grösse mit Word besser geklappt hätte. Da sich dies nicht auf die eingesetzte Zeit für die Implementation ausgewirkt hat, sondern nur den Gesamtaufwand für das Projekt vergrössert hat und ich zudem in diesem Bereich am meisten gelernt habe, würde ich es wohl wieder so machen. Könnte ich aber nochmal ein Projekt für die Studienarbeit wählen, würde ich mich aber nicht wieder für dieses Projekt entscheiden, dies vor allem Aufgrund des Einsatzes von *DICOM*.

9.2.4 Gelerntes

Wie bereits erwähnt, konnte ich bei diesem Projekt vor allem in den Bereichen um das Projekt herum am meisten profitieren. Hatte ich am Anfang noch fast keine Ahnung von \LaTeX , kann ich inzwischen damit fast alles bewerkstelligen, auch viele Sachen, die ich mit Word niemals umsetzen könnte. Des weiteren habe ich dank der schlechten Plugin-Unterstützung von CruiseControl auch das Ant-Scripting sehr ausführlich angewendet und bin nun dementsprechend sicher im Umgang damit, hatte ich doch vor Projektstart auch davon keine Ahnung. Aber auch im Bezug auf die Entwicklung von Embedded Devices konnte ich etwas dazulernen, denn bei der *GUI*-Entwicklung gibt es einen ganz anderen Ansatz, was teilweise zu Überraschenden Ergebnissen führt, wenn man sich damit nicht auskennt. Der Unterschied besteht nämlich darin, dass man keine Kontrolle über die Position von Widgets hat. Wie man damit Umgehen kann, konnte ich nun in dieser Arbeit erlernen.

9.2.5 Fazit

Trotz der vielen Probleme, die wir während der Arbeit hatte, bin ich mit dem Ergebnis zufrieden, da wir es trotzdem geschafft haben eine vorführbare Version zu erstellen mit welcher der Sinn und Zweck der Arbeit gezeigt werden kann. Bei der Bachelorarbeit werde ich mich aber wieder eher Richtung PC-Software orientieren.

9.3 Persönlicher Bericht: Yves Thrier

9.3.1 Einleitung

Nachfolgend werde ich die Studienarbeit revue passieren lassen und meine persönliche Sicht, meine Erfahrungen und meine Eindrücke dazu schildern: Was lief gut? Was könnte man besser machen? Wie zufrieden bin ich mit dem Ergebnis?

9.3.2 Die Studienarbeit

Der medizinische Informatikbereich war Neuland für mich. Ich empfand die Aufgabe und das Ziel als wichtiger Schritt in diesem Bereich und konnte mich gut damit identifizieren. Dies nahm auch die Scheu, sich intensiv in eine neue Thematik einzuarbeiten und wirkte sehr motivierend. Die Komplexität kam dann aber doch plötzlich überraschend und hat für viel «Haare raufen» gesorgt. Ich muss hier dazu einfach sagen, dass ich den *DICOM*-Standard als kaum mehr Zeitgemäss empfinde. Es sind viele – vermutlich Historisch bedingte – Lösungen für Funktionalitäten auf Arten beschrieben bzw. gelöst, die komplett Sinnfrei sind oder bei objektiver Betrachtung einfach keine Anwendung finden können. Auch der Aufbau des Standards in der Struktur und Gliederung ist verwirrend und kostet unnötig Zeit. Nun, da man sich mit den Problemen konkret und umfassend befasst hat, kann ich die spezifischen Punkte natürlich in kurzer Zeit erklären – dahinter stehen aber Stunden an Arbeit. Ich denke, dass die Thematik zwar durchaus für Studienarbeiten tauglich ist, unter der Voraussetzung das für einen Grossteil der Standardfunktionalität auf bestehende Lösungen zurück gegriffen werden kann. Eine eigene Implementation wie wir sie in dieser Arbeit getätigt haben, ist zwar möglich, aber steht in Anbetracht des Einarbeitungsaufwands in keinem Verhältnis zum Nutzen. Schlussendlich bin ich trotz allem Stolz auf das Ergebnis. Wir haben uns mit einem nicht sehr trivialen Problem auseinander gesetzt und dies trotz allen Schwierigkeiten noch «gemeistert», so dass ein «Endprodukt» verfügbar ist. Dazu muss man aber auch sagen, dass dieses Endprodukt nicht die geplante Wirkung erzielen kann oder wird. Ich sehe es als vielleicht eine art Beweis für die Realisierbarkeit, oder den Nutzen. Funktional ist der Umfang aber definitiv zu gering – als Ergebnis der zeitlichen Einschränkungen.

9.3.3 Rückblickend

Rückblickend auf die Studienarbeit ist für mich klar, dass dieses Themengebiet durchaus interessant ist, aber zuwenig Faszination ausstrahlt um beruflich darin tätig zu werden. Es hat sich an gewissen Stellen auch etwas Frustration eingestellt, da man sich jede kleine Information aus den Fingern saugen musste. Hätte man diese ganze Arbeit umsetzen wollen, hätte man auch bedeutend mehr Zeit benötigt. Dies hat sich aber nicht sehr früh heraus kristallisiert; Und die Semester sind nun mal nur 14 Wochen lang, so gibt es auch keinen Spielraum (dies ist wohl ein generelles Problem an solchen Arbeiten). Als sehr störend empfand ich auch kurzfristige Informationen über zusätzliche Arbeiten die «dann noch geleistet werden müssen» für die Abgabe. Sowas muss ganz klar ALLES von Anfang an feststehen, sonst kann man keine Zeitplanung machen.

9.3.4 Gelerntes

Bisher hatte ich keine Erfahrungen mit der Programmierung von Anwendungen für Smartphones o. ä. Diese nun gesammelte Erfahrung sehe ich als positiv an, obwohl ich sagen muss, dass ich kaum motiviert bin eine weitere Applikation für diese Endplattform zu schreiben. Die Mobiltechnologie steht in gewissen Bereichen noch stark in den Kinderschuhen, zum Leidwesen der Entwickler, aber verständlicherweise kann nicht alles von Heute auf Morgen funktionieren. Eine neue Erfahrung war es auch, ein Standard-Dokument nicht nur grob zu überfliegen, sondern auch das darin gelesene «konform» umzusetzen. Dieses intensive Auseinandersetzen (obwohl) mit einem Industrie-Standard empfinde ich als sehr wertvoll für meine weitere Laufbahn.

9.3.5 Fazit

Zusammenfassend war die ganze Arbeit sehr intensiv und stressig. Auch wenn das Ergebnis durchaus «erfolgreich» ist, bin ich andererseits doch auch ein wenig enttäuscht. Wir haben uns mit teilweise unnötigen Problemen herumgeschlagen, die im Nachhinein simpel hätten gelöst werden können; Nur diese Informationen waren nirgends vorhanden. Von daher bin ich froh, dass die Arbeit nun beendet ist, und bin motiviert neue Aufgaben in Angriff zu nehmen.

Literaturverzeichnis

- [And09a] ANDROID: Android Emulator (2009), URL <http://developer.android.com/guide/developing/tools/emulator.html>, [Online; Stand 4. Oktober 2009]
- [And09b] ANDROID: Android Market (2009), URL <http://www.android.com/market>, [Online; Stand 4. Oktober 2009]
- [ARP09] ARP: ARP DATACON - PC Onlineshop für Computer, Computerbedarf und Software (2009), URL <http://www.arp.ch>, [Online; Stand 3. Oktober 2009]
- [Cre09] CREATIVE COMMONS CORPORATION: Attribution-Share Alike 2.0 Generic (2009), URL <http://creativecommons.org/licenses/by-sa/2.0>, [Online; Stand 5. Oktober 2009]
- [dig09] DIGITEC.CH: digitec online shop (2009), URL <http://www.digitec.ch>, [Online; Stand 3. Oktober 2009]
- [dvt]
- [Gla06] GLATZ, Eduard: *Betriebssysteme*, dpunkt.verlag (2006)
- [GT09] GUHL, Mario und THRIER, Yves: *dvtk testdateien* (2009), URL http://www.dvtk.com/14_Testing/dvtk_testdateien
- [HDXY] HU, Zheng-wei; DUAN, Dong-xing; XIE, Zhi-yuan und YANG, Xing: Pipeline Design of Transformation between Floating Point Numbers Based on IEEE754 Standard and 32-bit Integer Numbers. URL <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4777556&isnumber=4777527>
- [Mic09] MICROSOFT: Microsoft Device Emulator 3.0 (2009), URL <http://www.microsoft.com/downloads/details.aspx?familyid=A6F6ADAF-12E3-4B2F-A394-356E2C2FB114&displaylang=en>, [Online; Stand 4. Oktober 2009]
- [Nat] NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION: URL <http://medical.nema.org>
- [Nat08a] NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION: *Part 5: Data Structures and Encoding* (2008), URL ftp://medical.nema.org/medical/dicom/2008/08_05pu.pdf
- [Nat08b] NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION: *Part 6: Data Dictionary* (2008), URL ftp://medical.nema.org/medical/dicom/2008/08_06pu.pdf
- [Nat08c] NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION: *Part 7: Message Exchange* (2008), URL ftp://medical.nema.org/medical/dicom/2008/08_07pu.pdf

- [Nat08d] NATIONAL ELECTRICAL MANUFACTURERS ASSOCIATION: *Part 8: Network Communication Support for Message Exchange* (2008), URL ftp://medical.nema.org/medical/dicom/2008/08_08pu.pdf
- [Nok09] NOKIA: Mobile J2ME Applications (2009), URL http://www.forum.nokia.com/Technology_Topics/Development_Platforms/Java.xhtml, [Online; Stand 4. Oktober 2009]
- [Pre09] PRENGEL, Frank: Windows Mobile 6.x – Intro für Entwickler (2009), URL http://www.techfiles.de/frankpr/Software%20f%FCr%20Windows%20Mobile%206%20entwickeln.htm#_Anwendungsarten, [Online; Stand 3. Oktober 2009]
- [Sam09] SAMSUNG: Samsung Mobile Innovator (2009), URL <http://innovator.samsungmobile.com/down/cnts/toolSDK.detail.view.do?platformId=2&cntsId=4604>, [Online; Stand 4. Oktober 2009]
- [Son09] SONY ERICSSON: Sony Ericsson Developer World (2009), URL <http://developer.sonyericsson.com>, [Online; Stand 4. Oktober 2009]
- [Sun99] SUN MICROSYSTEMS: Code Conventions for the Java Programming Language (1999), URL <http://java.sun.com/docs/codeconv/>
- [Tho01] THOUGHTWORKS, INC.: CruiseControl License (2001), URL <http://cruisecontrol.sourceforge.net/license.html>, [Online; Stand 5. Oktober 2009]
- [Top09] TOPPREISE.CH: Computer > Organizer (Preis/Preise/Preisvergleich Schweiz CH) (2009), URL http://www.toppreise.ch/grp2_127.html, [Online; Stand 3. Oktober 2009]
- [Vet09] VETTER, Florian: *Nachrüstbare Schnittstelle zur Einbindung von Diagnosegeräten in die medizinische IT-Infrastruktur*, Diplomarbeit, HSR (2009)
- [web09] WEBSHOP: Acer F900 (2009), URL http://www.enovo.ch/handys/acer/group_acer_f900, [Online; Stand 3. Oktober 2009]
- [Wik09a] WIKIPEDIA: Byte-Reihenfolge — Wikipedia, Die freie Enzyklopädie (2009), URL <http://de.wikipedia.org/w/index.php?title=Byte-Reihenfolge&oldid=67750596>
- [Wik09b] WIKIPEDIA: MIDlet — Wikipedia, Die freie Enzyklopädie (2009), URL <http://de.wikipedia.org/w/index.php?title=MIDlet&oldid=62724816>
- [Wik09c] WIKIPEDIA: Symbian OS — Wikipedia, The Free Encyclopedia (2009), URL http://en.wikipedia.org/w/index.php?title=Symbian_OS&oldid=318653885, [Online; accessed 13-October-2009]

-
- [Zei] ZEILINGER, Gunter: Open Source Clinical Image and Object Management, URL
<http://www.dcm4che.org/>

Glossar

Codestriker	Server-Tool zur Unterstützung von Codereviews: http://codestriker.sourceforge.net
DVTk	DVTk ist ein Open Source Projekt zum testen, validieren und diagnostizieren von Nachrichtenprotokollen und Szenarien in medizinischen Umgebungen: http://www.dvtk.org
Java Code Documentation	Werkzeug/Standard zur Java Code Dokumentation: http://java.sun.com/j2se/javadoc
MIDlet	Bezeichnet eine Software für ein Mobiltelefon, die in Java geschrieben ist, ähnlich einer jar=Datei auf einem PC, siehe auch [Wik09b].
Modalität	Eine Modalität ist ein Gerät in einer medizinischen Umgebung. Dies kann beispielsweise ein Röntgengerät oder Computertomograph sein.
Pre-testet commit	Möglichkeit neu eingeecheckte Dateien vor dem Einchecken auf Funktionstüchtigkeit zu prüfen
Task	Task ist der Überbegriff für die <i>DICOM</i> -Bezeichnung «planned procedure step», meint aber auch Vorgänge, die nicht an Modalitäten durchgeführt werden, wie z. B. ärztliche Untersuchungen. Ausserdem gilt ein Task auch dann noch, wenn der Vorgang schon abgeschlossen wurde, er kann also «planned» oder «proceeded» sein.
unsigned	Unsigned bedeutet «nicht vorzeichen behaftet», d. h. der Datentyp (eine Zahl) ist immer Positiv.

Abkürzungsverzeichnis

AD	Activity Diagram
BSD	Berkeley Software Distribution
C/S	Client/Server
CLDC	Connected Limited Device Configuration
CPD	Copy-Paste-Detector
CVS	Concurrent Versions System
DICOM	Digital Imaging and Communication in Medicine
ECTS	European Credit Transfer System
GUI	Graphical User Interface
HL7	Health Level 7
HSR	Hochschule für Technik, Rapperswil
HTML	Hypertext Markup Language
IDE	Integrierte Entwicklungsumgebung (engl. integrated development environment)
IEC	Internationale elektrotechnische Kommission (engl. International Electrotechnical Commission)
IHE	Integrating the Health Enterprise
ISO	Internationale Organisation für Normung (von gr. : «isos»; zu dt. «gleich»)
Jar	Java Archive
JCSC	Java Code Style Checker
JDK	Java Development Kit
JRE	Java Runtime Environment
KIS	Klinik-Informationssystem
LIS	Labor-Informationssystem
ME	Micro Edition
MIDP	Mobile Information Device Profile

NCSS	Non Commenting Source Statement
PDU	Protocol Data Unit
RIS	Radiologieinformationssystem
RUP	Rational Unified Process
SAD	System Architektur Dokument
SCP	Service Class Provider
SCU	Service Class User
SD	State Diagram
SDK	Software Development Kit
SOP	Service Object Pair
SSD	System Sequenz Diagramm
SVN	Subversion
TDD	Test Driven Development
UC	Use Case
UML	Unified Modeling Language
USB	Universal Serial Bus
VM	Virtual Machine
VPN	Virtual Private Network
VR	Value Representation
WTK	Wireless Toolkit

Tabellenverzeichnis

3.1	Übersicht Verantwortlichkeiten	8
3.2	Übersicht Meilensteine	9
3.3	Iterationsplanung	10
3.4	Wöchentliche Besprechungen	10
3.5	Übersicht Risiken	11
3.6	Eingetretene Risiken	15
3.7	Arbeitspakete	17
3.8	Übersicht Buildsysteme	19
3.9	Übersicht Bugtracker	19
3.10	Vergleich Zielplattformen	22
3.11	Angebote für Zielhardware	23
3.12	Usability Tests	26
3.13	Performance Tests	27
4.1	Stakeholders aus Kundensicht	43
4.2	Stakeholders aus Projektteamsicht	43
4.3	Stakeholders aus Endbenutzersicht	43
5.1	Association Request/Acknowledge Nachrichtenaufbau	75
6.1	Beispiel Implicit Codierung	93
6.2	Beispiel Explicit Codierung	93
6.3	Abstract Syntax Field	98
6.4	Application Context Field	98
6.5	Class UID Field	99
6.6	Implementation Class Field	99
6.7	Maximal Pdu Field	99
6.8	Presentation Context Field Request	100
6.9	Presentation Context Field Response	100
6.10	Transfer Syntax Field	100
6.11	User Information Context Field	101
6.12	Association Nachricht	101
6.13	Association Response	102
6.14	Association Release	102
6.15	Presentation Data Transfer	103
6.16	Presentation Data Value	103
6.17	Sequence Beispiel	109
7.1	Fragebogen Usability Tests	122
7.2	Fragebogen Usability Tests Corsin Camichel	123

7.3	Harte Grenzen Performance Test	123
7.4	Use Case Tests	124
7.5	Anforderungstests – Muss	124
7.6	Erfüllungsgrad	125
7.7	Codestatistiken	125
8.1	Vergleich Soll-/Ist-Resultat	132

Anhang A

Aufgabenstellung (Unterschrieben)

Aufgabenstellung Studienarbeit

„Workflowsteuerung und Bilddarstellung in Medizinischen Informationssystemen mit mobilen Clients“

1. Betreuer

Betreuer HSR:

- Axel Doering, Dozent für Informatik HSR

2. Ausgangslage, Problembeschreibung

Die Vernetzung von (Daten liefernden) Medizingeräten und medizinischen Informationssystemen nimmt seit einigen Jahren rasant an Bedeutung zu. Sie erlaubt die Verfügbarkeit verschiedenster Daten unabhängig vom Ort der Aufnahme ebenso wie die gezielte Steuerung des Arbeitsablaufs im komplexen Unternehmen „Arztpraxis“ oder „Spital“. Sie dient damit der Verbesserung der Behandlungsqualität und wird mittlerweile als der wesentlichste Faktor zur Effizienzsteigerung im Gesundheitswesen betrachtet.

Voraussetzung für eine Vernetzung von Geräten und Systemen unterschiedlichster Hersteller sind Kommunikationsstandards. DICOM (Digital Imaging and Communication in Medicine) und HL7 (Health Level 7) dienen dem 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.

Während in spezialisierten Fachabteilungen (Radiologie) die Auswertung von Bilddaten meist an speziellen, hochauflösenden, kalibrierten Befundungsstationen erfolgt, ist für viele Disziplinen (Kardiologie, Ophthalmologie, Nuklearmedizin) ein mobiler Zugriff auf Daten wichtiger als eine hohe grafische Darstellungsqualität. Dies gilt insbesondere für die Organisation des Arbeitsablaufs (welche Prozeduren sind mit welchen Patienten an welchen Geräten durchzuführen?) und die Erfassung von abrechnungsrelevanten Informationen (tatsächlich durchgeführte „procedure steps“, Verbrauchsmaterial).

3. Aufgabenstellung

Ziel der Arbeit ist die Entwicklung eines DICOM-Client für ein mobiles Gerät. Damit sollen in Kommunikation mit einem DICOM-Server Behandlungsaufträge („planned procedure steps“) für verschiedene Untersuchungsstationen („application entities“) abgefragt werden und tatsächlich durchgeführte Untersuchungen/Behandlungen („performed procedure steps“) zurückgemeldet werden.

Die Aufgabe kann dahingehend erweitert werden, dass auch Bilddaten vom mobilen Gerät aus abgefragt und

angezeigt werden können.

Damit Geräte, die unterschiedliche Teile des komplexen DICOM Standards umsetzen, zusammenarbeiten können, wurde der Meta-Standard IHE (Integrating the Health Enterprise) entwickelt. Systeme, die bestimmte Rollen innerhalb dieses Meta-Standards einnehmen wollen, müssen sich ausführlichen Tests unterziehen. Diese Tests sind, z.B. für die Rollen ADT Patient Registration, Image Display oder Report Reader, durchzuführen.

Voraussetzungen:

Die Bearbeitung der Aufgabe setzt eine Einarbeitung in die Grundlagen des DICOM Standards voraus, für die ca. 15 Arbeitsstunden anzusetzen sind. Für die Implementierung kann auf verschiedene Bibliotheken (z.B. JDICOM) zurückgegriffen werden.

Der DICOM Server muss nicht implementiert werden (er steht, basierend auf der open source Lösung www.dcm4che.org, aus einer vorangegangenen Studienarbeit zur Verfügung). Ebenfalls verfügbar sind diverse Testwerkzeuge für die IHE-Integration.

4. Zur Durchführung

Erwartet wird der Aufbau eines Buildsystems für einen kontinuierlichen Buildvorgang, bei dem gleichfalls Unittests (ggf. auch Integrations- und Systemtests) durchgeführt werden sollen.

Solide Programmierkenntnisse (vorzugsweise in Java) und die Bereitschaft, sich rasch neues Domänenwissen anzueignen, werden vorausgesetzt.

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 und dem Auftraggeber 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 Arbeitsergebnisse in vorläufigen Versionen abzugeben. Über die abgegebenen Arbeitsergebnisse erhalten die Studierenden ein vorläufiges Feedback. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Implementierung und Dokumentation.

5. Dokumentation

Über diese Arbeit ist eine Dokumentation zu verfassen gemäss den Richtlinien der Abteilung Informatik. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Die Dokumentation ist vollständig auf CD in drei Exemplaren abzugeben.

6. Termine

Siehe auch Terminplan auf <https://www.hsr.ch/Termine-Diplom-Bachelor-und.5142.0.html>

24.08.2009	Ausgabe der Aufgabenstellung durch den Betreuer
1. Semesterwoche	Kick-off Meeting an der HSR
3. Semesterwoche	Abgabe der Anforderungsanalyse
7. Semesterwoche	Abgabe Software-Architektur
8. Semesterwoche	Reviewmeeting Software-Architektur und Demonstration Prototyp
18.12.2009, 17:00 16. 	Abgabe der Arbeit an Betreuer Abgabe der Kurzbeschreibung an das Abteilungssekretariat

7. Beurteilung

Eine erfolgreiche Studienarbeit erhält 8 ECTS-Punkten (1 ECTS Punkt entspricht einer Arbeitsleistung von ca. 25 bis 30 Stunden). Für die Modulbeschreibung der Studienarbeit siehe

https://unterricht.hsr.ch/staticWeb/allModules/10938_M_SAI.html

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte (Abstract, Mgmt Summary, techn. u. persönliche Berichte) sowie Gliederung, Darstellung, Sprache der gesamten Dokumentation	1/5
3. Inhalt*)	3/5

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit mit dem Studenten vereinbart

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Studienarbeiten.

Rapperswil, den 17.09.2009

Der verantwortliche Dozent



Axel Doering

Professor für Informatik

Anhang B

Einrichtungsanleitung

B.1 Dokument

B.1.1 Dokumentinformationen

Version: 1.0
Revision: Rev: 416
Status: Release
Erstellt am: 12.12.2009
Erstellt von: Mario Guhl

B.2 Einführung

B.2.1 Zweck

Dieses Dokument beschreibt die Einrichtung einer funktionsfähigen Buildumgebung für dieses Projekt.

B.2.2 Gültigkeitsbereich

Diese Anleitung gilt für das gesamte Fortbestehen des Projekts und muss deshalb immer wieder nachgeführt werden, wenn sich an der Buildumgebung etwas ändert.

B.2.3 Übersicht

In den folgenden Sektionen wird beschrieben, welche Tools für das Projekt benötigt werden und wie diese installiert und konfiguriert werden müssen.

B.3 Java

Falls Java noch nicht auf dem System installiert sein sollte muss dieses als erstes installiert werden:

1. Java *JRE*¹ von der Seite <http://java.sun.com/javase/downloads/index.jsp> herunterladen.

¹ *Java Runtime Environment*

2. Installation des Java *JRE* starten und den Installationsanweisungen folgen.
3. Java *JDK*¹ von der Seite <http://java.sun.com/javase/downloads/index.jsp> herunterladen.
4. Installation des Java *JDK* starten und den Installationsanweisungen folgen.
5. Umgebungsvariable hinzufügen mit der Bezeichnung `JAVA_HOME` und als Wert den Installationspfad des *JDK* nehmen, normalerweise
`C:\Programme\Java\jdk1.6.0_16`.

B.4 Perl

Als nächstes muss Perl installiert werden, welches unter anderem für die Buildscripts benötigt wird:

1. Strawberry Perl von der Seite <http://strawberryperl.com> herunterladen.
2. Installer starten und den Installationsanweisungen folgen.
3. Normalerweise wird bei der Installation automatisch ein Eintrag in die `Path` Umgebungsvariable gemacht. Falls dies nicht der Fall ist muss das `bin` Verzeichnis von Strawberry Perl der `Path` Variable hinzugefügt werden, normalerweise
`C:\strawberry\perl\bin`.

B.5 IDE

B.5.1 eclipse

Als Nächstes muss die *IDE* installiert und konfiguriert werden:

1. Eclipse Galileo for Java von der Seite <http://www.eclipse.org/downloads> herunterladen (zweiter Eintrag).
2. In ein Verzeichnis nach Wahl entpacken.

B.5.2 S60 SDK

Nachdem eclipse installiert ist kann das *SDK*² für Symbian S60 von Nokia installiert werden:

1. Eclipse starten (workspace Ablage egal)
2. Im Menü `HELP`→`INSTALL NEW SOFTWARE...` wählen.
3. Bei `WORK WITH: --All Available Sites--` wählen.
4. Bei `type filter text` als Suchbegriff `Pulsar` angeben.
5. Den Eintrag `MOBILE AND DEVICE DEVELOPMENT`→`ECLIPSE PULSAR` markieren.

¹ *Java Development Kit*

² *Software Development Kit*

6. Auf NEXT klicken und den Installationsanweisungen folgen.
7. Nach dem Installieren eclipse neu starten.
8. Im Menü WINDOW→SHOW VIEW→OTHER... wählen, dann PULSAR→MOBILE SDKS auswählen und OK klicken.
9. Im Register MOBILE SDKS den Eintrag FORUM NOKIA SDK→S60→S60_5TH_EDITION_SDK_v1_0_EN.ZIP auswählen und auf den grünen + Button klicken (oben rechts)
10. Der Installationsanleitung folgen (Herunterladen kann eine Weile dauern, da das Paket über 600 MB gross ist).
11. Wenn ein weiterer Installationsdialog erscheint sollte ein Installationspfad ohne Leerzeichen angegeben werden, allfällige Perl Fehler/Warnungen am Ende der Installation können ignoriert werden. Der Device-Command Dialog kann ebenfalls ignoriert werden.
12. Irgendwann muss der eclipse Installationspfad angegeben werden, dort muss der Hauptpfad der eclipse Installation angegeben werden.
13. Am Ende der Installation wird ein Dialog angezeigt, in dem ein Emulator ausgewählt werden muss, dort auf SELECT MANAGED DEVICES... klicken.
14. Danach IMPORT... wählen und mittels BROWSE... den Installationspfad des S60 SDKs auswählen, normalerweise C:\S60.
15. Nach einem Klick auf OK wird der Ordner durchsucht und es sollten zwei Devices gefunden werden; S60Emulator und S60Device.
16. Nach einem Klick auf FINISH den S60Emulator markieren und OK drücken.
17. Danach nochmals den S60EMULATOR auswählen und REMEMBER THIS MATCH ebenfalls markieren, danach nochmals OK drücken.

B.5.3 Mobile Tools for Java

Damit die Software für das Mobiltelefon kompiliert werden kann, werden noch die «Mobile Tools for Java» benötigt:

1. Eclipse starten (workspace Ablage egal) falls noch nicht gestartet.
2. Im Menü HELP→INSTALL NEW SOFTWARE... wählen.
3. Bei WORK WITH: Galileo auswählen.
4. Bei type filter text als Suchbegriff mobile angeben.
5. Den Eintrag MOBILE TOOLS FOR JAVA und MOBILE TOOLS FOR JAVA SDK markieren.
6. Auf NEXT klicken und den Installationsanweisungen folgen.
7. Nach dem Installieren eclipse neu starten.

B.6 Buildscript Tools

Damit die Buildscripts funktionieren müssen noch einige Tools installiert werden.

B.6.1 Java WTK

Das Java *WTK*¹ wird benötigt um die Testabdeckung zu ermitteln und bietet ausserdem einen schlanken Emulator an, welcher oftmals anstatt dem langsamen Nokia Emulator benutzt werden kann. Das *WTK* muss folgendermassen installiert werden:

1. Java *WTK* von der Seite <http://java.sun.com/products/sjwtoolkit> herunterladen.
2. Installation des Java *WTK* starten und den Installationsanweisungen folgen.
3. Umgebungsvariable hinzufügen mit der Bezeichnung `WTK_HOME` und als Wert den Installationspfad des *WTK*, normalerweise `C:\WTK2.5.2_01`.
4. Umgebungsvariable hinzufügen mit der Bezeichnung `WTK_FILESYSTEM` und als Wert `C:\Documents and Settings\\j2mewtk\2.5.2\appdb\DefaultColorPhone\filesystem` bzw. ab Windows Vista `C:\Users\\j2mewtk\2.5.2\appdb\DefaultColorPhone\filesystem`.

B.6.2 Open Office

Wird benötigt um Officedokumente in das PDF-Format umzuwandeln:

1. Open Office von der Seite www.openoffice.org herunterladen.
2. Installation starten und den Installationsanweisungen folgen.
3. Den Pfad `<Open Office Installationspfad>\program` der Path Umgebungsvariable hinzufügen.

B.6.3 MiKTeX

Dieses Tool wandelt die \LaTeX Dokumente in das PDF-Format um:

1. MiKTeX von der Seite <http://miktex.org> herunterladen.
2. Installation starten und den Installationsanweisungen folgen.
3. Während der Installation `INSTALL MISSING PACKAGES ON-THE-FLY` auf `YES` setzen.

B.7 Dokumentationswerkzeuge

Für die komfortable Erstellung der \LaTeX Dokumente werden noch weitere Tools benötigt. Zum Erstellen der Dokumente werden diese aber nicht benötigt, dies ist mit den Buildscripts bereits möglich.

¹ *Wireless Toolkit*

B.7.1 PDF-XChange PDF Viewer

Wenn der Acrobat Reader bereits installiert ist, wird der PDF-XChange PDF Viewer nicht mehr unbedingt benötigt, allerdings integriert er sich besser in das TeXnicCenter.

1. PDF-XChange PDF Viewer von der Seite <http://pdf-xchange-viewer.softonic.de> herunterladen.
2. Installer starten und bei der Serialabfrage die freie Version auswählen, ausser man besitzt diese bereits.

B.7.2 TeXnicCenter

Wird benötigt um die \LaTeX Dateien zu bearbeiten. Dies geht zwar auch mit einem normalen Editor, ist aber mit dem TeXnicCenter wesentlich komfortabler.

1. Den TeXnicCenter Installer von der Seite <http://www.texniccenter.org> herunterladen.
2. Installation starten und den Installationsanweisungen folgen.
3. TeXnicCenter starten und Tool Tip schliessen.
4. Im Konfigurationsassistenten auf NEXT klicken. Den Pfad zum Mi \TeX bin Verzeichnis angeben, normalerweise `C:\Program Files\Mi \TeX 2.8\miktex\bin` und zwei mal auf NEXT klicken.
5. Den Pfad zum PDF-XChange PDF Viewer angeben, normalerweise `C:\Program Files\Tracker Software\PDF Viewer\PDFXCview.exe`.
6. Im zweiten Textfeld `/close "%bm.pdf"` eingeben und im dritten Textfeld `"%bm.pdf"`, beides mit Anführungszeichen.
7. Auf NEXT und dann auf FINISH drücken.
8. Im TeXnicCenter Menü AUSGABE→AUSGABEPROFILE DEFINIEREN... auswählen.
9. Auf der linken Seite LaTeX => PDF auswählen.
10. Rechts bei PFAD DES MAKEINDEX-COMPILERS: `makeindex.exe` ersetzen durch `makeglossaries.exe`.

B.8 Tortoise SVN

11. Tortoise SVN von der Seite <http://tortoisesvn.net/downloads> herunterladen.
12. Installation starten und den Installationsanweisungen folgen.
13. Danach kann das Projekt ausgecheckt werden. Das Checkout Verzeichnis sollte maximal 30 Zeichen lang sein und keine Leerzeichen enthalten, da einige Tools damit Mühe haben.

Anhang C

Bedienungsanleitung

C.1 Dokument

C.1.1 Dokumentinformationen

Version: 1.0
Revision: Rev: 407
Status: Release
Erstellt am: 16.12.2009
Erstellt von: Mario Guhl

C.1.2 Dokumenthistory

Rev.	Datum	Wer	Änderung
0	16.12.2009	ythrier	Dokument erstellt.

C.2 Zweck

Dieses Dokument ist die Bedienungsanleitung der Software Ihe4Mobile. Die Bilder können von der hier gezeigten leicht abweichen. Bei anderen Geräten können sie je nachdem stark davon abweichen. Die dargestellten Informationen sind aber immer gleich.

C.3 Benutzung

C.3.1 Hauptansicht

Die Hauptansicht wird beim Starten der Applikation angezeigt. Beim ersten Starten wird eine Aufforderung zur Konfiguration der Verbindungs- und Autorisierungsinformationen angezeigt (siehe C.3.2). Von der Hauptansicht kann auch über `OPTIONS`→`TASKLIST` auf die Suchmaske für die Arbeitsschritte gewechselt werden.

C.3.2 Settings

Hier wird Adresse des *DICOM*-Servers, dessen Port und ein Passwort für die Datenverschlüsselung konfiguriert. Ausserdem das Intervall in dem ein Report automatisch gespeichert wird.

C.3.3 Arbeitsschritte Suchmaske

Auf der Suchmaske für die Arbeitsschritte kann anhand von Patientendaten (Name/Vorname) und einer *Modalität* nach Arbeitsschritten gesucht werden. Die Suche wird mittels des SEARCH gestartet.

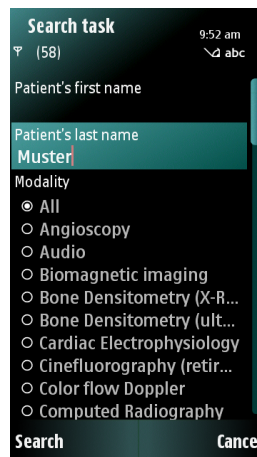


Abbildung (C.1) Suchmaske für die Arbeitsschritte

C.3.4 Liste der Arbeitsschritte

Von der Suchmaske wird man bei erfolgreicher Suche auf die Liste der gefundenen Arbeitsschritte weiter geleitet. Hier werden alle Schritte dargestellt, die der Suchanfrage entsprechen. Durch einen Doppelklick bzw. durch Drücken von **OPTIONS**→**DETAILS** können für einen Arbeitsschritt die Details angezeigt werden.

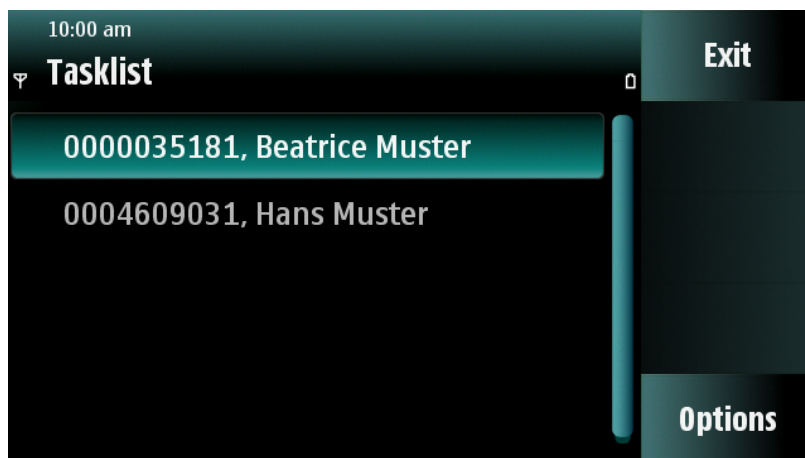


Abbildung (C.2) Liste der Arbeitsschritte

C.3.5 Details eines Arbeitsschrittes

Hier wird die Identifikationsnummer und das Startdatum des Schrittes angezeigt sowie die betroffene *Modalität*. Am Ende der Details kann ein Report hinzugefügt werden. Ebenfalls ist es möglich, mittels Selektion des Patientennamens dessen Details anzuzeigen (alternativ über OPTIONS→DETAILS). Unter OPTIONS kann der Schritt lokal als erledigt markiert werden. Dies geschieht mittels COMPLETE.



Abbildung (C.3) Details eines Arbeitsschrittes

C.3.6 Details eines Patienten

Hier werden die wichtigsten Patienteninformationen angezeigt, wie Adresse, Telefonnummer, Gewicht/Grösse und Kommentare.

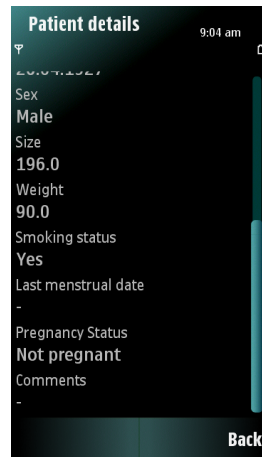


Abbildung (C.4) Details eines Patienten

C.3.7 Report

Hier kann über ein Bericht geschrieben werden. Dieser wird über SAVE gespeichert. Ausserdem wird er automatisch in einem bestimmten Intervall (siehe C.3.2 auf Seite 163) gespeichert.

Anhang D

Zeitplan

Zeitplan SAI

Phasen Meilensteine Woche Task	Inception						Elaboration (1. Iteration)						Elaboration (2. Iteration)														
	Total		Woche 1		Woche 2		Woche 3		Woche 4		Woche 5		Woche 6														
	SOLL	IST	14.09 yt	18.09 mg	21.09 yt	25.09 mg	28.09 yt	02.10 mg	05.10 yt	09.10 mg	12.10 yt	16.10 mg	19.10 yt	23.10 mg													
1. Projektmanagement	52.0	161.0	11.0	11.0	19.0	19.0	10.0	12.5	10.0	18.5	0.0	2.0	2.0	16.5	0.0	0.0	0.0	22.5	0.0	0.0	0.0	24.0	0.0	0.0	0.0	7.0	
1.1 Projektplan	20.0	37.5	5.0	5.0	5.0	5.0	5.0	6.5	5.0	7.5	0.0	0.0	0.0	22.5	0.0	0.0	0.0	2.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	7.0	
1.2 Verwaltung / Infrastruktur	10.0	72.0	2.0	2.0	4.0	4.0	2.0	0.5	2.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1.3 Qualität	3.0	3.5					3.0	3.5																			
1.4 Review / Anpassungen	2.0	4.5	1.0	1.0			1.0	1.5																			
1.5 Risk Management	4.0	4.0	3.0	3.0			1.0	1.0																			
1.6 Dokumentvorlagen	13.0	39.5			10.0	10.0			3.0	10.5																	
2. Anforderungen	42.0	24.5	0.0	0.0	0.0	0.0	4.0	2.0	4.0	0.5	12.0	10.5	10.0	6.5	6.0	3.0	6.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0
2.1 Use Cases «brief»	10.0	2.2					2.0	1.0	2.0	0.2	2.0	0.0	2.0	1.0	1.0	0.0	1.0										
2.2 Use Cases «fully dressed»	12.0	5.5									4.0	0.0	4.0	3.5	2.0	0.0	2.0					2.0					
2.3 Supplementary Specifications	6.0	10.0									4.0	8.0			2.0	2.0											
2.4 Anforderungskatalog	10.0	3.8					2.0	1.0	2.0	0.3	1.0	0.5	3.0	2.0	1.0	0.5	3.0	2.0									
2.5 Review Anforderungen	4.0	3.0					1.0	2.0	1.0		1.0	2.0	1.0		1.0	1.0	1.0										
3. Analyse	34.0	30.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	6.0	7.0	6.0	1.5	8.0	6.0	8.0	0.0	4.0	0.0	2.0	5.0	
3.1 User Environment Model	4.0	0.0													2.0	0.0			2.0	0.0							
3.2 Domänenmodell	5.0	10.5									2.0	5.0	2.0	1.5	2.0	5.0	2.0	1.5	2.0	1.0							1.0
3.3 Contracts	6.0	5.0															3.0				3.0						2.0
3.4 System Sequence Diagram	4.0	3.0									1.0	0.0			1.0	0.0			2.0	2.0					1.0	0.0	
3.5 State / Activity Diagramm	2.0	4.0													2.0	0.0			2.0	0.0							2.0
3.6 Externes Design	7.0	4.0																			2.0				3.0	0.0	2.0
3.7 Review	6.0	4.0									1.0	2.0	1.0		2.0	2.0	2.0		2.0	2.0	2.0						
4. Internes Design	53.0	22.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0	7.0	7.0	0.0	13.0	5.0	13.0	5.0	
4.1 Klassendiagramm	16.0	8.0													4.0	3.0	4.0		4.0	3.0	4.0		2.0	1.0	2.0	4.0	
4.2 Kommunikationsarchitektur	12.0	4.0													4.0	4.0	3.0										3.0
4.3 Internes GUI Design	4.0	0.0																									4.0
4.4 Logische Architektur (Schichtenmodell)	7.0	6.0																									3.0
4.5 Problem Domäne	8.0	4.0																									4.0
4.6 Review	6.0	0.0																									2.0
5. Implementation	131.0	239.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	5.0	20.0	5.0	0.0	
5.1 Prototypen (GUI, PD, Architektur, C/S)	24.0	46.0																		4.0							2.0
5.2 Automation	5.0	9.0																									5.0
5.3 Datenhaltung	3.0	9.5																									2.0
5.4 Fehlerbehandlung	7.0	6.0																									
5.5 Review und Refactoring	20.0	34.5																									
5.6 Projekt / Codestruktur	7.0	10.0																									
5.7 Ausbau GUI	24.0	8.5																									
5.8 Ausbau Problem Domäne	41.0	115.5																									
6. Test	55.0	49.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6.1 Unit Tests	24.0	39.0																									
6.2 Usability Tests	8.5	0.0																									
6.3 Performance Test	4.5	0.0																									
6.4 Fehlerbehebung	10.0	10.0																									
6.5 System Test	8.0	0.0																									
7. Dokumentation	84.0	86.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7.1 Anleitung	10.0	5.0																									
7.2 Software Architecture Document	48.0	27.0																									
7.3 Abschluss	12.0	30.0																									
7.4 Abschlussbericht	10.0	21.0																									
7.5 Erfahrungsreport	4.0	3.0																									
Tasks wiederkehrend																											
8. Technologiestudium	44.0	44.0	6.0	6.0	0.0	0.0	5.0	5.0	5.0	0.0	5.0	5.0	5.0	2.0	5.0	13.0	5.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8.1 DICOM	44.0	44.0	6.0	6.0			5.0	5.0	5.0		5.0	5.0	5.0		5.0	13.0	5.0		5.0								
9. Sitzungen	39.0	19.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0	2.0	2.0	1.0	1.0	1.0	1.0	2.0	1.5	2.0	0.0	1.0	1.0	1.0	1.0	0.0	2.0	0.0
9.1 Sitzungsprotokoll	13.0	3.0							1.0		1.0	1.0							1.0	0.5							
9.2 Besprechung mit Betreuer	26.0	16.0	1.0	1.0	1.0	1.0					1.0	1.0	1.0	1.0	1.0	1.0	1.0		1.0						1.0	1.0	
10. Qualitätssicherung	22.0	14.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0
10.1 Reviews (Dokumentation / Code-Style)	22.0	14.0									1.0		1.0						1.0		1.0						
Wochen Subtotal			18.0	18.0	20.0	20.0	19.0	19.5	20.0	19.0	20.0	19.5	19.0	26.0	18.0	24.0	19.0	25.5	19.0	22.0	17.0	27.0	23.0	25.0	22.0	17.0	
Wochentotal	556.0	689.0			38.0	38.0	39.0	38.5	39.0	45.5	37.0	49.5	36.0	49.0	45.0	42.0											

Auswertung	GESAMT		yt		mg		Differenz (Ist-Soll)		Differenz Akkumuliert
	SOLL	IST	SOLL	IST	SOLL	IST	yt	mg	
Inception	77.0	7							

Anhang E

Betreuer-Meetings

Betreuer-Meeting

Datum: 29.09.2009

Protokollführer: Yves Thrier

Traktandenliste

1. Besprechung Projekt- und Zeitplan (10')
2. Verbesserungen Projekt- und Zeitplan (5')
3. Review der Anforderungen (20')
4. Vorschläge für weitere Anforderungen und Use Cases (15')
5. Diverses (5')

Zeiten sind variabel.

Protokoll

1. Projektplan + Zeitplan
 - a. Genauere Besprechung 6.10.09
 - b. Deliverables besser definieren
2. Anforderungen / UC
 - a. Worklist
 - b. Daten holen
 - c. Service Personal / Medizinisches Personal / Service Personal / Administratives Personal
 - d. Externe Aktoren
3. HW Beschaffung im Verlauf Donnerstag 1.10 zuschicken

Beschlüsse

1. Genaueres definieren der Deliverables
 - a. Anleitung, Abstract
 - b. ...
2. Kein Worklist management
3. Keine Daten erfassen (im Sinne von „Bildern“, aber Diagnosen denkbar)
4. Korrektur und hinzufügen von Befunden
5. Cache „fast“ zwingend (Verkaufs (Noten ☺) Argument

Diverses

1. DICOM Structured Report -> Genauer anschauen
2. IHE Workflow und Roles

Betreuer-Meeting

Datum: 06.10.2009

Protokollführer: Mario Guhl

Traktandenliste

1. Feedback Projekt- und Zeitplan (15')
2. Hardwarebeschaffung (10')
3. Review der Anforderungen und Use Cases (20')
4. Weitere Schritte (10')
5. Diverses (5')

Zeiten sind variabel.

Protokoll

1. Feedback Projekt- und Zeitplan
 - a. In der Construction Phase 3 sollte ebenfalls ein Performance Test eingeplant werden
 - b. Testfälle müssen in Kategorien eingeteilt werden, die beschreiben, wie schwerwiegend es ist, wenn ein Test fehlschlägt
 - c. Bei Zielhardware sollte auch die Laufzeit der vorgeschlagenen Smartphones aufgezeigt werden
 - d. Projektplan sonst in Ordnung
 - e. Herr Doering hat Bedenken, ob das Projekt überhaupt in der Zeit schaffbar ist
 - f. Eingeplante Zeit für Externes Design ist etwas knapp
 - g. Zeitplan sonst ausgeglichen und in Ordnung
2. Hardwarebeschaffung
 - a. Als Zielhardware wurde das Nokia N97 bestimmt, da es beim Android Dev Phone bedenken gab, ob es in nützlicher Zeit lieferbar ist, da es in den USA bestellt werden muss
 - b. Es wurde festgehalten, dass man bei einem Projekt mit wirtschaftlichem Interesse aus Kostengründen das HTC Dreams genommen hätte
3. Review der Anforderungen und Use Cases
 - a. Use Case View Patient Data: Es sollte auch möglich sein Informationen über den Patienten (Name, Krankheitsgeschichte usw.) anzuzeigen
 - b. Abschnitt Richtigkeit sollte genauer definiert werden: z.B. Verhalten bei Fehlerhaften Daten oder Widersprüchen definieren
 - c. Abschnitt Fehlertoleranz/Wiederherstellbarkeit sollte genauer definiert werden: z.B. im Falle eines Verbindungsunterbruchs sollte der Benutzer darüber informiert werden, welche Daten schon an den Server geschickt wurden
 - d. Es sollte ein Planungsdokument für die Tests erstellt werden
4. Weitere Schritte
 - a. Mario Guhl: Buildumgebung aufbauen
 - b. Yves Thrier: Prototyp

Beschlüsse

1. Performance Test in Construction Phase 3 einplanen
2. Testfälle in Kategorien einteilen
3. Bei Zielhardware Laufzeit aufzeigen
4. Für Externes Design mehr Zeit einplanen

5. Use Case View Patient Data um Anzeige von Patienteninformationen erweitern
6. Abschnitt Richtigkeit genauer definieren
7. Abschnitt Fehlertoleranz genauer definieren
8. Abschnitt Wiederherstellbarkeit genauer definieren
9. Planungsdokument erstellen

Diverses

1. Die Links zu dem Build-Server sollen an Herr Doering weitergeleitet werden, wenn der Build-Server eingerichtet ist
2. Darauf achten, dass der Code der zum ausprobieren erstellt wurde nicht plötzlich als Grundlage für die Zielsoftware verwendet wird

Betreuer-Meeting

Datum: 13.10.2009

Protokollführer: Yves Thrier

Traktandenliste

1. Domain Model (15')
2. CruiseControl (10')

Zeiten sind variabel.

Protokoll

1. Fehlen noch punkte im DomainModel / Verbindung / Korrelation zu DM entities
2. Serie <-> ProcedureStep. Unterscheidung zum DICOM Procedure step
3. Definition Begriff in Kontrast zu DICOM Begriffen (Glossary event. Aufnehmen) -> Trennung / Unterschied

Beschlüsse

1. EA Reverse engineering dcm4che2
2. MS / Termine sind okay (ende woche)

Diverses

1. Montag vormittag spätestens material für Dienstag abend
2. Ende woche (Donnerstag abend):
 1. Projektplan
 2. Zeitplan / Aufwandsabschätzung

Betreuer-Meeting

Datum: 27.10.2009

Protokollführer: Mario Guhl

Traktandenliste

1. Probleme mit JavaME und dcm4che2
2. Feedback Domain Analyse
3. Feedback Projektplan Update
4. Feedback Zeitplan Update
5. Diverses

Zeiten sind variabel.

Protokoll

1. Probleme mit JavaME und dcm4che2
 - a. Nach Schilderung der Probleme, ist Herr Doering damit einverstanden, dass die bisherige Mussanforderungen Patienten suchen/anzeigen in eine optionale Anforderung umgewandelt wird, und dafür Teile der dcm4che2-Library für JavaME neu geschrieben werden (siehe auch Projektplan→Risikomanagement→Eingetretene Risiken)
 - b. SDDs usw. in der Domain Analyse die die neu optionalen Anforderungen betreffen können weggelassen werden
2. Feedback Domain Analyse
 - a. Herr Doering ist mit DomainAnalyse insbesondere mit dem DomainModel überhaupt nicht zufrieden, da der Geschäftszweck nicht ersichtlich ist.
 - b. Nach Besprechung mit Herr Doering stellt sich heraus, dass es evt. gar nicht möglich ist, das DomainModel besser zu machen, da sich die Logik hauptsächlich auf dem Server abspielt.
 - c. Herr Doering schlägt vor, dass Domain-Modell einem Mitstudenten zu zeigen und es so zu belassen, wenn er das Domain-Modell versteht
 - d. Herr Doering gibt dem Team eine Frist bis 01.11.2009 23:59 um die Domain Analyse zu überarbeiten
3. Feedback Projektplan/Zeitplan /Anforderungsspezifikation
 - a. Das E-Mail mit den entsprechenden Dokumenten ist bei Herr Doering untergegangen, deshalb wird er uns bis zum Wochenende dazu ein Feedback geben
5. Diverses
 - a. Herr Doering schlägt vor eine Dokumenthistory einzuführen, damit schnell ersichtlich ist, was geändert wurde, wenn ein Dokument überarbeitet wurde.
 - b. Die Analysetools die in den automatischen Buildvorgang integriert sind gefallen Herr Doering sehr gut.

Beschlüsse

1. Domainanalyse bis 01.11.2009 23:59 überarbeiten
2. Dokumenthistory einführen
3. Muss/Sollanforderungen in der Anforderungsspezifikation überarbeiten

Diverses

1. Die Links zu Bugzilla, Codestriker und Analysetools an Herr Doering weiterleiten

Anhang F

Team-Meetings

Team-Meeting

Datum: 06.10.2009

Protokollführer: Yves Thrier

Traktandenliste

1. Arbeitsverteilung
2. dcm4che2
3. Build-Server
4. Meeting 17.15 – 18.15
5. Probleme
6. Diverses

Protokoll

1. Modellierungsaufgaben gegen Ende Woche, vorher dcm4che2 überlegen wie das genau funktioniert und tiefer einarbeiten (Modellieren Samstag @ Rapperswil)
 1. Yves: dcm4che2
 2. Mario: Buildserver und Produkte
2. Siehe 1)
3. Verlauf dieser und nächster Woche
4. Produktwahl und Java ME: Wird nicht explizit hingeschrieben. Diskussion effekt. Endgerät heute Abend.
5. -
6. -

Weiteres Vorgehen

Wer	Was	Erledigt
Yves	Dcm4che2	Ok
Mario	Buildserver	Ok
Mario	Produktwahl Verbesserung (Lebensdauer, Investitionskosten)	Ok
Yves	Rechtschreibung / Review Anforderungen (TeXnicCenter) Dict.)	50%
Yves + Mario	Modellieren / Analysetask W4 (Samstag)	Ok
Yves	NFR (Richtigkeit, Fehlertoleranz, Wiederherstellbarkeit)	Ok
Yves	DICOM Validation und "Proof of Functionality and Testing"	Ok
Yves	Performance Test für Construction 3 W2	Ok
Yves	Kategorie Testfälle (Gewichtung)	-
Mario	UC View Patient Data (Erweitern um Anzeige Patientendaten / neuer UC)	Ok

Team-Meeting

Datum: 29.10.2009

Protokollführer: Yves Thrier

Traktandenliste

1. Domainanalyse Dokument (ToDo's)
2. Architektur
3. Library Portierung
4. Risikomanagement
5. Anforderungsspezifikationen

Protokoll

1. Domainmodel anderer Person zeigen (S. Hunkeler)
2. SSDs zu UC (zu denen die auch gemacht werden nach Anpassung)
3. SSD zu Verbindungsaufbau (Associated, Reject, Abort,...)
4. Use Case anpassen (auch Procedure Step zu Task)
5. GUI Prototyp (Zeichnen Visio)
6. SSD Strukturierung (UC verweis, namentlich, o.ä.)
7. Contracts Review (nach SSD Überarbeitung)
8. State Diagramme
9. User Environment model
10. Package description (Einzeln und Schnittstellen)
11. Review Klassendiagramm
12. Funktionsbeschreib
13. Risikomanagement Eintrag bzg. Aufwand (Portierung)
14. Anforderungsspezifikation Muss -> Optional -> Change history / Sitzung

Weiteres Vorgehen

Wer	Was	Erledigt
Yves Thrier	Domain Model mit entity description Sebastian zeigen ob er es so verstehen würde	Ok (s.u.)
Yves Thrier	SSD zu notwendigen Use Cases	Ok
Mario Guhl	Use Case Anpassungen (Procedure Step -> Task und Verbindungs-UC zu Verbindungseinstellungs UC)	Ok
Yves Thrier	SSD Verbindungsaufbau (StateDiagramm sinnvoll?)	Ok
Mario Guhl	SSD Strukturierung (siehe Protokoll)	Ok
Mario Guhl	GUI Prototyp zeichnen	Ok
Mario Guhl	Package Description (ausser Network) und Review CD	Ok
Yves Thrier	Package Description Network und CD dazu	Ok
Yves Thrier	User Environment Model	Ok
Y.T + M.G.	Contracts Review	Ok(mguhl)
Yves Thrier	Logische Schichten (Optional)	Ok (inkl. Descr.)
Yves Thrier	Funktionsbeschreib Library Portierung (in Domain Analyse mit	Ok

	rein nehmen)	
Mario Guhl	Risikomanagement Eintrag	Ok
Mario Guhl	Anforderungsspezifikationsänderung (s.o.)	Ok

Diskussion Domainmodel mit S. Hunkeler und C. Ramseier:

Ihrer Meinung nach, kann man so oder so nicht den Ablauf der Business Logik in einem Domainmodel erkennen, es gehe mehr darum, die Konzepte, die daran teilnehmen abzubilden (was der Fall ist). Die Funktionalität werde anhand der Use Cases und der Sequenzdiagramme klar. Lediglich eine bessere Anordnung der Entitäten helfe eventuell das Augenmerk mehr auf die Service Entitäten zu legen.

Team-Meeting

Datum: 03.11.2009

Protokollführer: Yves Thrier

Traktandenliste

1. Aufgabenverteilung Heute
2. Meeting Heute
3. Testing

Protokoll

1. Diskussion DICOM Network
2. GUI erste Versuche machen
3. SAD Tasks dieser Woche
4. Testing Construction 1

Weiteres Vorgehen

Wer	Was	Erledigt
M.G + Y. T.	DICOM network diskussion	Ok
M. Guhl	GUI Erste implementierung	Ok
Y. Thrier	SAD Network	Ok
M. G. + Y. T.	SAD review tasks diese woche	Ok
Y. Thrier	Testing dokument construction 1	-
M. G. + Y. T.	Tests Construction 1 durchführen	-
M. Guhl	SAD Business Layer	Ok
M. G + Y. T	Leistung + bereits mögliche Punkte einfügen	Ok
Y. Thrier	Dcm4che Server aufsetzen (Nach F. Vetter SAI)	-

