HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
INFORMATIK

INSTITUT
FÜR
SOFTWARE

Demonstrator für die elektronische Versichertenkarte

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Frühjahrssemester 2010

Autoren: Michael Hofmann, Mark Lowe
Betreuer: Prof. Dr. Axel Doering
Experte: Christian Kohler
Gegenleser: Prof. Dr. Josef Joller

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

Kurzfassung der Bachelorarbeit

Abteilung	**Informatik**
Name[n] der Studierenden	**Michael Hofmann, Mark Lowe**
Studienjahr	**[FS 2010]**
Titel der Studienarbeit	**Demonstrator für die elektronische Versichertenkarte**
Examinatorin / Examinator	**Prof. Dr. Axel Doering**
Themengebiet	**Software**
Projektpartner	
Institut	**Institut für Software**

Kurzfassung

Zielsetzung:

Ziel dieser Bachelorarbeit war es, einen Demonstrator für die elektronische Versichertenkarte der Schweiz zu entwickeln. Dieser sollte eine Teilmenge von Daten und Funktionen gemäss eCH-0064-Standard zu Schulungszwecken visualisieren. Ferner sollte ein Recherchebericht zum aktuellen Entwicklungsstand rund um das System Versichertenkarte verfasst, sowie Schulungsunterlagen zu Versichertenkarte und Demonstrator erarbeitet werden.

Umfeld:

Die Schweizerische Versichertenkarte ist Teil der Strategie eHealth Schweiz und wird seit Anfang 2010 von den Versicherungsgesellschaften ausgeliefert. Sie baut auf SmartCard-Technologie auf und bietet gemäss Spezifikation die Möglichkeit, neben administrativen Daten auch medizinische Notfalldaten zu speichern. Der Zugang zu den medizinischen Notfalldaten ist jedoch geschützt und kann ausschliesslich über ein Card-To-Card-Authentifizierungsverfahren mit Hilfe eines Leistungserbringerausweises (HPC) freigeschaltet werden.

Ergebnisse:

Im Rahmen der Bachelorarbeit wurde ein Demonstrator entwickelt, der spezifikationskonforme Daten von realen Versichertenkarten aller Hersteller gemäss eCH-0064-Standard visualisieren kann. Zudem wurde als Teil des Demonstrators ein wiederverwendbares API entwickelt , welches auch in andere Projekte integriert werden kann. Ferner wurden ein Recherchebericht sowie Schulungsunterlagen verfasst.

Fazit:

Der entwickelte Demonstrator kann einerseits für Schulungszwecke eingesetzt werden und andererseits auch als Basis für eine spätere, produktive Implementierung dienen.

Inhaltsverzeichnis

Teil 3: Hardware-Beschaffung

Teil 4: Software-Projektdokumentation

Anforderungen

Analyse

Design

Implementierung

Statistiken

Installation

Projektmanagement

Zeitplanung

Glossar

Demonstrator für die elektronische Versichertenkarte

Management Summary

1 Management Summary

1.1 Ausgangslage

Der Bundesrat hat 2006 die Strategie für eine Informationsgesellschaft in der Schweiz aus dem Jahr 1998 revidiert. Einer der beiden neuen Schwerpunkte (neben dem E-Government) ist der Einsatz von Informations- und Kommunikationstechnologien im Gesundheitswesen, kurz als eHealth bezeichnet. 2007 wurde dazu ein Strategiepapier veröffentlicht, welches als Hauptziel die Einführung eines elektronischen Patientendossiers vorgibt. Als ersten konkreten Schritt in diese Richtung, hat das Schweizer Parlament im Oktober 2004 mit Artikel 42a im Bundesgesetz vom 18. März 1994 über die Krankenversicherung (KVG; SR 832.10) die rechtliche Grundlage für die Einführung einer Versichertenkarte geschaffen, die sowohl administrative- als auch persönliche Daten elektronisch speichern kann. Seit dem 1. Januar 2010 werden diese Versichertenkarten von den Versicherern ausgestellt.

Abbildung 1: Versichertenkarten und HPC

1.2 Vorgehen / Technologien

Ziel der Bachelorarbeit war die Entwicklung eines Demonstrators für die Verwendung der elektronischen Versichertenkarte, die Erarbeitung einer Übersicht zum Entwicklungsstand der Versichertenkarte und das Erstellen von Schulungsunterlagen zu Demonstrator und elektronischer Versichertenkarte. Wichtige Kriterien für die Entwicklung des Demonstrators waren eine möglichst hohe Plattformunabhängigkeit und die Möglichkeit, den Demonstrator sowohl Standalone, als auch in eine Website eingebunden, verwenden zu können. Basierend auf diesen Kriterien wurde das System in Java entwickelt. Der Zugriff auf die Versichertenkarte erfolgt über ein selbst entwickeltes API, welches wiederum auf dem Java Smart Card I/O API aufbaut. Die grafische Benutzerschnittstelle wurde in JavaFX implementiert. Weiter wurde eine Website für Schulungszwecke und Download des Demonstrators entwickelt.

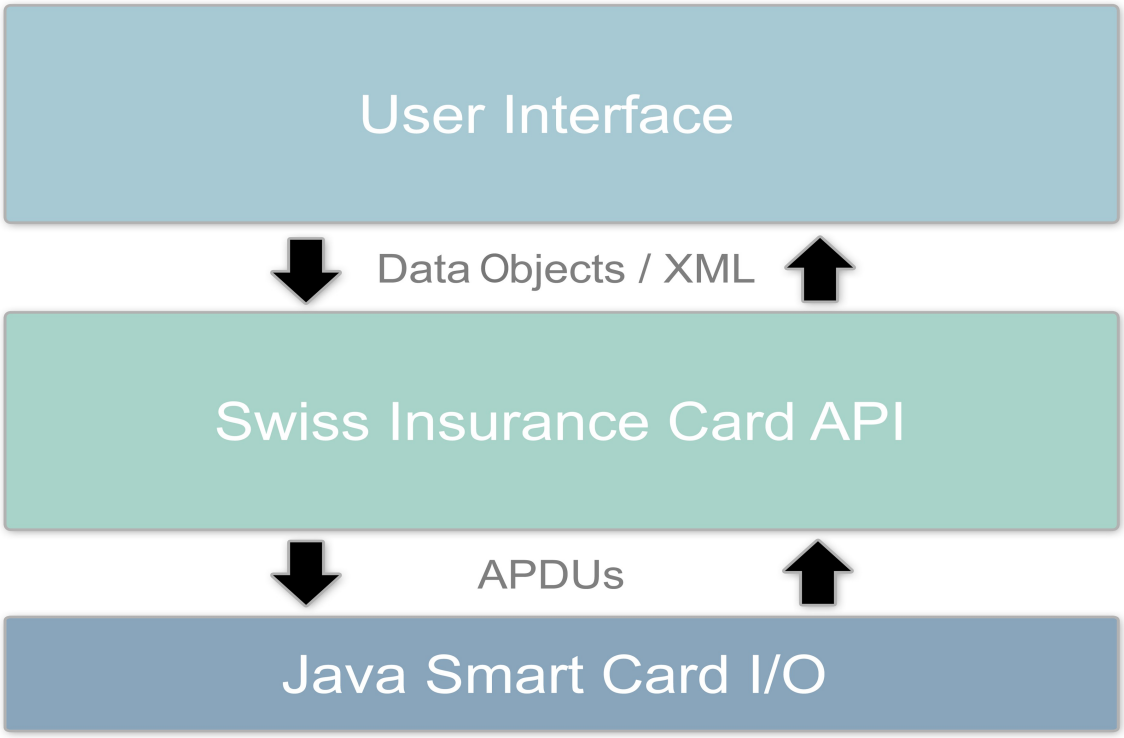


Abbildung 2: Architekturübersicht

1.3 Ergebnisse

Mit dem Demonstrator können die Daten aller spezifikationsgemässen Versichertenkarten nach Standard eCH-0064 ausgelesen werden. Auch entsprechende Authentifizierungsmechanismen wurden implementiert. Der Demonstrator kann sowohl mit, als auch ohne grafische Benutzeroberfläche verwendet werden. Mit der grafischen Benutzeroberfläche kann man sich die von der Versichertenkarte gelesenen Daten anzeigen lassen und auch komplexere Transaktionen durchführen. Das für den Demonstrator entwickelte API kann auch direkt in bestehende oder künftige Projekte, für den Zugriff auf Daten und Funktionen der elektronischen Versichertenkarte, eingebunden werden. Dabei kann auf verschiedenen Abstraktionsebenen gearbeitet werden.

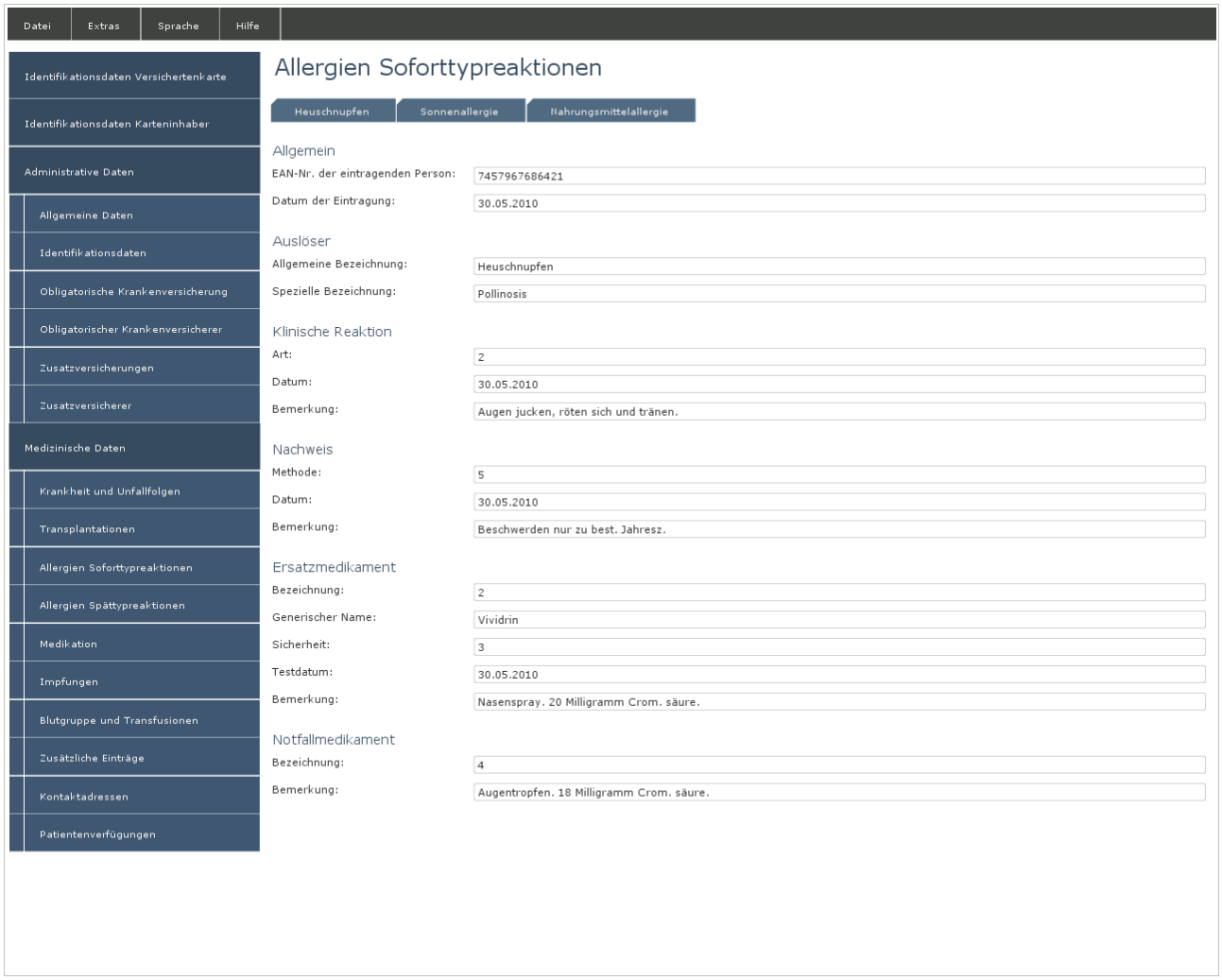


Abbildung 3: Screenshot: Anzeige der medizinischen Daten

Elektronische Versichertenkarte

Technischer Bericht

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-03-27
Geändert	2010-06-18
Autoren	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einführung

1.1 Problemstellung, Vision

Der Bundesrat hat 2006 die Strategie für eine Informationsgesellschaft in der Schweiz aus dem Jahr 1998 revidiert. Einer der beiden neuen Schwerpunkte (neben dem E-Government) ist der Einsatz von Informations- und Kommunikationstechnologien im Gesundheitswesen, kurz als eHealth bezeichnet. 2007 wurde dazu ein Strategiepapier veröffentlicht, welches als Hauptziel die Einführung eines elektronischen Patientendossiers vorgibt. Als ersten konkreten Schritt in diese Richtung, hat das Schweizer Parlament im Oktober 2004 mit Artikel 42a im Bundesgesetz vom 18. März 1994 über die Krankenversicherung (KVG; SR 832.10) die rechtliche Grundlage für die Einführung einer Versichertenkarte geschaffen, die sowohl administrative- als auch persönliche Daten elektronisch speichern kann. Seit dem 1. Januar 2010 werden diese Versichertenkarten von den Versicherern ausgestellt.

1.2 Aufgabenstellung

1.2.1 Aufgabenstellung vom Betreuer

„Ziel der Arbeit ist zunächst die Erarbeitung einer Übersicht zum Entwicklungsstand der Versichertenkarte (welche Anbieter, welche Referenzimplementierungen, welche Features werden derzeit unterstützt, in welche Infrastruktur werden Versichertenkarten derzeit eingebunden). Weiterhin soll eine Experimentierumgebung entwickelt werden, die im Minimum die Pflichtfelder einer spezifikationsgemässen Versichertenkarte auslesen kann. Diese Funktion soll so implementiert werden, dass sie über einen Webbrowser verfügbar gemacht werden kann (Anschluss eines geeigneten Cardreaders vorausgesetzt). Optional ist eine prototypische Implementierung eines geeigneten Featuresubsets gemäss eCH-0064 denkbar, so dass auch Schreiboperationen und komplexere Transaktionen getestet werden können. Das Entwicklungsergebnis soll als Demonstrator für die Lehre eingesetzt werden.

Eine weitere Option ist die Entwicklung und Testung spezifischer SmartCard – Funktionen unabhängig vom Einsatz als Versichertenkarte, wie z.B. Verwendung als Signatkarte und zur Ver- und Entschlüsselung, und damit möglicherweise auch für Micropayment-Funktionen.“

1.2.2 Ziele und Unterziele

Hauptziel der Arbeit war die Entwicklung eines Demonstrators für den Einsatz in der Lehre, der im Minimum die administrativen Daten einer spezifikationsgemässen Versichertenkarte auslesen kann. Dazu gehörte einerseits die Einarbeitung in SmartCard Grundlagen und andererseits das Studium der relevanten Standards rund um das System Versichertenkarte. Ebenso musste die benötigte Hardware (Kartenlesegerät, Testkarten) evaluiert und beschafft werden. Zudem war eine Schulungsdokumentation zum Thema elektronische Versichertenkarte zu erstellt.

Optional konnten weitere Features implementiert werden, so dass auch Schreiboperationen und komplexere Transaktionen getestet werden können. Eine weitere Option war der Test von weiteren SmartCard-spezifischen Funktionen, unabhängig vom Einsatz als Versichertenkarte. (Siehe Aufgabenstellung)

1.3 Vorgehen, Aufbau der Arbeit

Bereits früh wurde in einer Recherche der aktuelle Entwicklungsstand rund um das System Versichertenkarte in der Schweiz erfasst. Dabei stellte sich rasch heraus, dass die physische Auslieferung der Versichertenkarte gerade erst begonnen hatte (Anfang 2010) und dass es zwei Hersteller gab. Dies war einerseits die Schweizerische Post[1], welche im Auftrag der Versicherungsgesellschaft Helsana (grösster Schweizer Versicherer[2]) Karten herstellte. Andererseits war das die Firma SASIS[3], die im Auftrag von Santésuisse (Verband der Schweizerischen Krankenversicherer) die Karten für alle anderen Versicherungsgesellschaften herstellte. Weiter stellte sich heraus, dass die beiden Kartenimplementierungen, die beide dem Standard eCH-0064 folgten, trotzdem gewisse Unterschiede aufwiesen. Der grösste Unterschied hierbei stellten die verwendeten Zertifikate dar, die bei der Post-Implementierung 1024 Bit Schlüssellänge aufwiesen während bei der SASIS-Karte die Schlüssellänge 2048 Bit betrug. Auf Grund dieser Problematik und auch auf Grund der Tatsache, dass die Einführung der Versichertenkarte gerade erst begonnen hatte, gab es zum Zeitpunkt der Recherche noch keine ausgereifte Software, die das Lesen / Schreiben der medizinischen Notfalldaten auf der Versichertenkarte ermöglicht hat. Eine Referenzimplementierung oder ein API war auch noch nicht vorhanden.

Ebenfalls in der Anfangsphase der Bachelorarbeit wurde die benötigte Hardware evaluiert. Hier wurden

1 Vgl. http://www.post.ch/healthcare/ (Abrufdatum: 12. Juni 2010)
2 Vgl. http://www.helsana.ch (Abrufdatum: 12. Juni 2010)
3 Vgl. http://www.veka-center.ch (Abrufdatum: 12. Juni 2010)

Kriterien an ein Kartenleseterminal aufgestellt, welches die Arbeit mit eCH-0064-konformen SmartCards erlaubt hat. Ausserdem mussten die nötigen Testkarten von den Herstellern angefordert werden. Das waren einerseits die beiden bereits erwähnten Kartenhersteller sowie die FMH (Schweizerische Ärztevereinigung), die für die Herausgabe der Leistungserbringerausweise (HPC) zuständig war. Seitens der FMH zeigte man sich interessiert an einer möglichen OpenSource Lösung, da zu dem Zeitpunkt noch keine Referenzimplementierung vorhanden war.

Basierend auf diesen Erkenntnissen wurde die Sachlage neu beurteilt und entschieden, dass weiterhin ein Demonstrator für Schulungszwecke entwickelt werden soll, der über eine wiederverwendbare API-Schicht auf die Kartendaten- und Funktionen zugreift. Ein Lesen / Schreiben von medizinischen Daten auf der Karte bestand weiterhin als optionales Ziel - dies war jedoch vom Erhalt von signierten HPC Zertifikaten seitens der FMH abhängig.

Direkt nach Erhalt der ersten Testkarte wurde ein Prototyp entwickelt, der ein Auslesen der administrativen Daten auf einer SASIS-Kartenimplementierung möglich machte. Danach wurde der Demonstrator in zwei Teilen (Benutzeroberfläche, Kartenzugriff) mit vordefinierten Schnittstellen entwickelt und in einer späteren Phase zusammengeführt. Näheres hierzu siehe Abschnitt 3 Umsetzungskonzept.

Parallel zur Entwicklung wurden Schulungsunterlagen erarbeitet, die zusammen mit dem Demonstrator für Schulungszwecke eingesetzt werden können.

1.3.1 Risikofaktoren

Als den grössten Risikofaktor in diesem Projekt, wurde der Erhalt von signierten HPC-Zertifikaten identifiziert. Auf Grund der Neuheit der Materie und dem Umstand, dass sich noch nicht viele Personen mit der Thematik auseinandergesetzt haben, bestand auch das Risiko unvollständiger, unverständlicher oder fehlerhafter Dokumentationen, sowie Implementierungsfehler seitens der Kartenhersteller.

1.3.2 Involvierte Personen

Involviert an der Arbeit waren die beiden Studenten Michael Hofmann und Mark Lowe, die von Prof. Dr. Axel Doering betreut wurden. Weitere Personen haben unterstützend zu diesem Projekt beigetragen. (Details siehe Abschnitt 5 Dank)

1.4 Übersicht über weitere Teile der Arbeit

Name	*Beschreibung*
Technischer Bericht	Dieses Dokument. Weitere Kapitel: Stand der Technik, Umsetzungskonzept, Resultate.
Rechercheberich t	Bericht zum Entwicklungsstand rund um das System „Versichertenkarte“.
Software-Projektdokumentation	Anforderungsspezifikation, Analyse, Design, Implementation, Test, Projektmanagement, Projektmonitoring.

Table 1: Übersicht über weitere Teile der Arbeit

2 Stand der Technik

2.1 Bestehende Lösungsansätze und Normen

Da sich die Einführung der Versichertenkarte in der Schweiz noch in einem frühen Stadium befand, existierten zum Zeitpunkt der Bachelorarbeit noch keine Referenzimplementierungen oder sonstige Lösungsansätze. Es konnte einzig auf den Angaben im eCH-0064-Standard sowie in den publizierten Detailspezifikationen von Post[4] und SASIS[5] aufgebaut werden. Des weiteren musste für die Implementierung der Kartenfunktionen auf die Angaben im Standard ISO/IEC-7816-4[6] für SmartCards sowie auf Angaben aus dem Standard ISO/IEC 9796-2[7] für Kryptologieverfahren zurückgegriffen werden. Das Bundesamt für Gesundheit BAG hat zudem XML Schemata veröffentlicht[8], welche die Zusammensetzung der auf der Versichertenkarte speicherbaren administrativen und medizinischen Daten definierten.

Wie bereits erwähnt, war zum Zeitpunkt der Bachelorarbeit noch keine Software verfügbar, die sowohl das Lesen von administrativen Daten, als auch das Lesen / Schreiben medizinischer Daten von beiden Kartenherstellern unterstützte. Die für die Authentifizierung notwendigen HPC-Zertifikate waren seitens FMH noch nicht verfügbar.

Für die Post-Testkarten waren jedoch selbst-signierte Testzertifikate vorhanden, die allerdings nur mit den von der Post herausgegebenen Musterkarten funktionierten. Eine allfällige Nutzung dieser Testzertifikate bedingte allerdings, dass Kryptologische Funktionen des HPC-Kartenbetriebssystems softwareseitig simuliert werden mussten.

4 Vgl. „Implementierungsanleitung für die Versichertenkarte nach eCH-0064 v.1.0“ von SwissSign
5 Vgl. „Aktuelle Detailspezifikationen SASIS - Versichertenkarte nach VVK 832.105 v. 1.3“ von SASIS
6 Vgl. „ISO 7816 Part 4: Organization, security and commands for interchange“
7 Vgl. „ISO 9796 Part 2: Integer factorization based mechanisms“
8 Vgl. http://www.bag.admin.ch/themen/krankenversicherung/07060/index.html?lang=de (12. 06. 2010)

3 Umsetzungskonzept

3.1 Grundidee

Grundidee der neu entwickelten Software waren die automatische Erkennung der Kartenimplementierung (Post / SASIS / FMH), sobald eine Karte in das Kartenlesegerät eingefügt wird, die Umwandlung der Rohdaten auf den Versichertenkarten in spezifikationskonforme Datenobjekte und als dritter Punkt, der Aufbau einer Benutzerschnittstelle auf diesen Funktionen - zur Demonstration der vorhandenen Daten und Funktionen.

Die entwickelte Software sollte dabei in ein eigenständiges API-Paket sowie in eine darauf aufbauende Benutzerapplikation unterteilt werden. Das API sollte dabei als einfach zu benutzende Schnittstelle zwischen den Kartenfunktionen / Kartendaten und einer darauf aufbauenden Applikation dienen. Diese Applikation soll vom API Datenobjekte oder XML-Daten erhalten, ohne sich um Implementierungsdetails der Versicherten-karten kümmern zu müssen.

Die darauf aufbauende Benutzerapplikation sollte zur Verwendung für Schulungszwecke dienen und die vorhandenen Daten und Funktionen der angeschlossenen Versichertenkarte(n) demonstrieren.

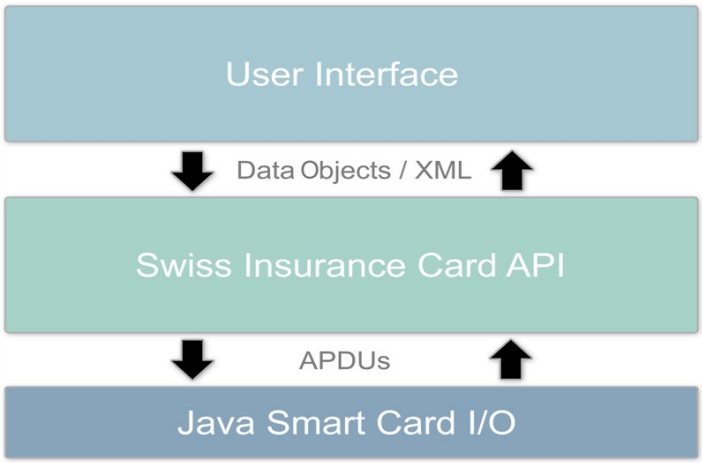


Illustration 1: Konzeptionelle Architekturübersicht

Der Demonstrator (API + Benutzerschnittstelle) soll möglichst Plattformunabhängig und ohne Installation lauffähig sein, direkt im Browser oder Standalone als Download.

3.2 Verwendete Technologien

Für die Umsetzung wurden verschiedene Technologien evaluiert. Einerseits wurde hier nach bestehenden Lösungen für den Zugriff auf SmartCard-Funktionen gesucht, andererseits wurde auch eine geeignete Technologie für die Implementierung der Benutzerschnittstelle evaluiert.

3.2.1 SmartCard Zugriff

Zur Evaluation von SmartCard Zugriffsbibliotheken wurde der folgende Anforderungskatalog aufgestellt:

Kriterium	*Beschreibung*
Nutzbar ohne Installation auf Client-System	Die Nutzung des Demonstrators soll ohne Installation auf dem Client-System möglich sein.
Spezifische Smart Card Kommandos ermöglichen	In den Spezifikationen der Kartenhersteller sind Funktionen sehr spezifisch dokumentiert (konkrete APDU-Befehle). Diese müssen 1:1 gesendet werden können.
Unterstützung von Standard-Kartenlesegeräten	Die Unterstützung von gängigen Kartenlesegeräten soll möglich sein.
Unterstützung von ISO-7816 Smart Cards	Das Lesen und Durchführen von Operationen auf ISO-7816- sowie eCH-0064-konformen SmartCards soll möglich sein.
Geringer Einarbeitungsaufwand	Auf Grund der engen Zeitspanne des Projekts, sollte die Nutzung der Bibliothek mit geringem Einarbeitungsaufwand verbunden sein.
OpenSource / Lizenzfrei	Da das Entwicklungsergebnis später möglicherweise einmal als OpenSource-Lösung veröffentlicht werden könnte, sollte die zu verwendende Schnittstelle uneingeschränkt nutzbar sein und idealerweise öffentlich zugänglichen Quellcode haben.
Entwicklungsstand / Verbreitung / Funktionsumfang	Der Entwicklungsstand, die Verbreitung, sowie die Reife der Bibliothek soll in Betracht gezogen werden.

Table 2: Anforderungen an die zu verwendende Kartenzugriffs-Bibliothek

Die folgenden Bibliotheken / Frameworks wurden evaluiert:

- Opencard Framework / OpenSCDP[9]
- Opensc mit PKCS#11[10]
- Java SmartCard I/O API[11]

Ergebnisse der Evaluation:

Kriterium	*Opencard Framework*	*Opensc mit PKCS#11*	*Java SmartCard I/O API*
Installation nötig	Teilweise (.dll / .so)	Ja, .dll / .so	Nein, in JRE integriert
Spezifische Kommandos	Ja	Nein, nur Zertifikat-Zugriff	Ja
Unterstützung Lesegeräte	Ja	Ja	Ja
Unterstützung ISO-7816	Ja	Teilweise	Ja
Einarbeitungsaufwand	Mittel	Mittel	Gering, da nicht komplex
OpenSource / Lizenzfrei	Spezielle Lizenz	Ja	Ja
Entwicklungsstand	Wird nicht mehr weiter entwickelt.	Wird weiter entwickelt.	Teil des Java Runtime Environment
Funktionsumfang	++	- (nur Zertifikats-Funktionen)	- (nur Grundfunktionen)
Verbreitung			

Table 3: Vergleich der Kartenzugriffs-Bibliotheken

Auf Grund der Evaluation wurde das Java SmartCard I/O als Grundlage für die weitere Entwicklung ausgewählt. Gegenüber dem Opencard Framework musste zwar mehr Funktionalität selbst entwickelt werden, dafür bot es grössere Flexibilität.

Die Verwendung von Java SmartCard I/0 war ein weiterer Grund, neben der Plattformunabhängigkeit, für die Wahl von Java als Entwicklungssprache.

3.2.2 Benutzerschnittstelle

Für die Benutzerschnittstelle musste ebenfalls eine geeignete Technologie gefunden werden. Folgende Anforderungen sollten erfüllt werden:

Kriterium	*Beschreibung*
Unterstützung von Java	Da der Kartenzugriff über Java implementiert wird, sollte die Benutzerschnittstelle ebenfalls in Java implementiert werden.
Lauffähig in Webbrowser	Die Applikation sollte optimalerweise direkt in einem Browserfenster ausführbar sein.

Table 4: Anforderungen an die zu verwendende Benutzerschnittstellen-Technologie

Die folgenden Benutzerschnittstellen-Technologien wurden evaluiert:

- Swing
- JavaFX

Mit beiden Technologien war es möglich, diese direkt in einem Browserfenster auszuführen. Zugriff auf die Karten-API war ebenfalls mit beiden möglich. Da JavaFX prädestiniert ist für plattformübergreifende, im Web lauffähige Applikationen mit intuitiver Benutzeroberfläche, wurde JavaFX, aufgrund dieser Überragenden Fähigkeiten, als Basis für die Entwicklung der Benutzerschnittstelle ausgewählt.

9 http://www.openscdp.org/ocf/ (Stand 13.06.2010)
10 http://www.opensc-project.org/ (Stand 13.06.2010)
11 http://java.sun.com/javase/6/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-tree.html (Stand 13.06.2010)

4 Resultate

4.1 Demonstrator

Im Rahmen der Bachelorarbeit wurde ein Java-basierter Demonstrator für Schulungszwecke entwickelt, der die administrativen Daten von Versichertenkarten beider Schweizer Hersteller auslesen und anzeigen kann. Die Anzeige von medizinischen Daten wurde vorbereitet und kann mit virtuellen Karten simuliert werden.

Der Demonstrator erkennt angeschlossene Versichertenkarten und HPCs automatisch und zeigt diese an, sobald sie in ein Kartenlesegerät eingefügt werden. Erkannte Karten können ausgewählt und deren Daten angezeigt werden. Die angezeigten Daten können in Schema-konforme XML-Dateien exportiert werden. Ein Import von Daten aus XML-Dateien ist ebenfalls möglich. Des weiteren wurden PIN-Code-Verwaltungsfunktionen implementiert.

4.2 API

Als Teil des Demonstrators wurde ein eigenständiges API für eCH-0064-konforme Versichertenkarten und HPCs entwickelt, das auf der Java SmartCard I/O API aufbaut. Es bietet eine einfach zu benutzende Schnittstelle für den Zugriff auf Kartenfunktionen- und Daten.

Funktionalität:

- Automatische Erkennung des Kartenherstellers, gesonderte Handhabung von Zertifikaten und herstellerspezifischen Kartenbefehlen
- Vereinfachte Lese- und Schreibfunktionen unabhängig vom Kartenhersteller
- Automatische Erkennung von angeschlossenen Karten und Lesegeräten
- Multithreading-fähige Änderungserkennung (z.B. bei Entfernen/Hinzufügen einer Karte)
- Umwandlung von codierten Kartendaten in spezifikationskonforme Datenobjekte oder XML
- Card-to-Card-Authentifizierung nach eCH-0064 für den Einsatz mit realen Zertifikaten vorbereitet
- Simulation des Card-to-Card-Authentifizierungsverfahrens auf HPC-Seite für den Einsatz mit Post-Testkarten vorbereitet
- Logging- und Debugging-Funktionen
- Kartenbefehl-Emulator für automatisiertes Testen

Das Implementieren von Lese / Schreiboperationen auf medizinischen Daten von realen Versichertenkarten konnte nicht implementiert werden, da die dazu nötigen Zertifikate von der FMH nicht geliefert werden konnten. Das API ist jedoch vorbereitet für eine spätere Arbeit mit realen Karten und Zertifikaten.

4.3 Schulungsunterlagen

Es wurden Schulungsunterlagen verfasst, die einen Überblick über SmartCard- und Kryptographie-Grundlagen, sowie über die eCH-0064 Spezifikation geben. Zudem wurde die Funktionsweise des Demonstrators erläutert. Diese Dokumentation wurde zusammen mit dem Demonstrator in eine Website integriert.

4.4 Recherchebericht

Ein Recherchebericht zum Entwicklungsstand rund um das System Versichertenkarte wurde verfasst.

4.5 Ausblick / Weiterentwicklung

Der Demonstrator wurde primär für Schulungszwecke entwickelt. Das API kann auch direkt in bestehende oder künftige Projekte integriert werden.

Folgende Punkte aus dem eCH-0064-Standard könnten in künftigen Projekten noch implementiert werden:

- Card-To-Card Authentifizierung mit realen Karten
- Lesen und Schreiben von medizinischen Daten
- Online-Abfrage gemäss eCH-0064

Bei einer Weiterentwicklung des API wäre zu prüfen, inwiefern sich dieses parallel zu anderen Karten-Diensten betreiben lässt. Die HPC der FMH kann z.B. auch für die Signierung von E-Mails verwendet werden. Die Kryptographie-Dienste des E-Mail-Clients müssten daher ebenfalls auf die Kartenzertifikate zugreifen, was über eine PKCS#11-Implementierung geschieht. Ebenso muss der parallele Betrieb von Applikationen der Kantonalen Modellversuche sichergestellt werden, da auch diese auf Daten der Versichertenkarte zugreifen.

Weitere, detailliertere Informationen hierzu können der Softwaredokumentation im Abschnitt „Weiterentwicklung“ entnommen werden.

5 Dank

Für die fachliche Unterstützung im Verlauf dieser Bachelorarbeit bedanken wir uns bei:

Prof. Dr. Axel Doering, für die Unterstützung und Betreuung unserer Bachelorarbeit.

Dr. Harald Heuser, für die umfassenden Informationen und Beschaffung der Testkarte.

SwissSign AG, für die Informationen zur Versichertenkarte der Post und die Beschaffung der Testkarten.

SASIS AG, für die Informationen zu ihrer Versichertenkarte und die Beschaffung einer Testkarte.

FMH, für die Bereitstellung einer HPC für Developer.

6 Anhang

6.1 Abbildungsverzeichnis

6.2 Tabellenverzeichnis

Elektronische Versichertenkarte

Rechercheberichte

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-03-27
Geändert	2010-06-18
Autoren	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Im Oktober 2004 wurde mit Artikel 42a vom Schweizer Parlament die Grundlage zur Einführung des Systems „Versichertenkarte“ geschaffen. Dieser Bericht umfasst den aktuellen Entwicklungsstand, per Ende März 2010, rund um die Versichertenkarte in der Schweiz. Es werden die politische Situation, Auslieferungsstand, involvierte Hersteller und der aktuelle technische Entwicklungsstand beschrieben.

1.2 Zweck

Dieser Bericht dient dazu, einen Überblick über den aktuellen Entwicklungsstand der Versichertenkarte in der Schweiz zu erhalten.

1.3 Gültigkeit

Dieses Dokument gilt zum aktuellen Zeitpunkt Ende März 2010. Da der Einführungsprozess der Versichertenkarte zur Zeit noch in Gange ist, kann sich der Stand der Dinge jederzeit ändern.

1.4 Vorgehen

Der aktuelle Entwicklungsstand rund um das System „Versichertenkarte“ wurde durch Internet-Recherche und direktes Anfragen beim BAG (Bundesamt für Gesundheit) und den beteiligten Parteien (Verbände, Hersteller) ermittelt. In den Bericht eingeflossen sind Publikationen des BAG, Rundschreiben an die Versicherer und veröffentlichte Sitzungsprotokolle sowie weitere öffentlich zugängliche Informationen. Es wurden zudem technische Spezifikationen des Bundes und Detailspezifikationen der Hersteller studiert und als „proof of concept“ eine Software entwickelt, mit der administrative Daten von einer Versichertenkarte ausgelesen werden können.

2 Beteiligte Parteien / Stakeholder

Am System „Versichertenkarte“ gibt es eine Reihe verschiedener Stakeholder.

2.1 Bundesamt für Gesundheit BAG

Das BAG koordiniert die Einführung der Versichertenkarte und kontrolliert die Einhaltung der vom Bund herausgegebenen Spezifikationen.

2.2 Versicherte

Alle der obligatorischen Krankenversicherung unterstehenden Personen sollen eine neue Versichertenkarte erhalten. Die versicherten Personen sind an Datensicherheit und Transparenz der auf ihrer Chipkarte gespeicherten Daten interessiert.

2.3 Versicherungsgesellschaften

Die Versicherungsgesellschaften werden von der Santésuisse, dem Dachverband der Schweizer Krankenversicherer, vertreten. Sie sind für die Herstellung und Auslieferung spezifikationskonformer Versichertenkarten verantwortlich. Diese Aufgabe wurde an verschiedene in Abschnitt 4 genannte Hersteller delegiert.

2.4 Berufsverband der Schweizer Ärzte (FMH)

Die Interessen der Ärzte werden von der Verbindung der Schweizer Ärztinnen und Ärzte (FMH) vertreten. Die FMH stellt eine Ärzte-Ausweiskarte (HPC) her, die eine eindeutige Identifikation von Leistungserbringern erlaubt. Die Ärzte sind an einer effizienten Abrechnung mittels Versichertenkarte interessiert, haben sich jedoch vermehrt kritisch gegenüber dem Speichern von medizinischen Daten auf der Chipkarte geäussert.[1]

2.5 Spitäler

Die Spitäler werden vom Verband H+ vertreten. Sie sind interessiert an einer effizienten Abrechnung mittels Versichertenkarte, teilen jedoch die Meinung der FMH zu den medizinischen Daten.

2.6 Apotheker

Die OFAC (Berufsgenossenschaft der Schweizer Apotheker) und die Pharmasuisse vertreten die Interessen der Apotheker. Sie sind interessiert an einer effizienten Abrechnung mittels Versichertenkarte. Diese beiden Verbände führen zudem das „Schweizer Medikamentdendossier“ ein.[2]

2.7 Kantone

Kantone haben die Möglichkeit, im Rahmen von Modellversuchen, die weiterführende Nutzung der Versichertenkarte zu untersuchen.
„Im Rahmen von kantonalen Modellversuchen im Gesundheitsbereich ist die erweiterte Nutzung der Versichertenkarte über den Zweck von Artikel 42a Absatz 2 KVG und über die Nutzungsmöglichkeiten nach Artikel 42*a* Absatz 4 KVG hinaus möglich, sofern dies im kantonalen Recht vorgesehen ist.“[3]

2.8 Hard- und Softwarehersteller

Hard- und Softwarehersteller wollen mit der Versichertenkarte kompatible Lösungen anbieten können.

1 Vgl. Sitzungsprotokoll „Weiteres Vorgehen Versichertenkarte – Runder Tisch“ 3.12.2009
2 Vgl. Pressemitteilung OFAC „Gegensätzliche Debatte über die Zukunft der Versicherten- und Gesundheitskarte“ 26.9 2008
3 VVK Art. 16 Abs. 1

3 Politische Situation

Die rechtliche Grundlage für die Einführung des Systems „Versichertenkarte“ wurde im Oktober 2004 mit Artikel 42a im Krankenversicherungsgesetz [KVG] geschaffen. Die Verordnung vom 14. Februar 2007 über die Versichertenkarte für die obligatorische Krankenpflegeversicherung [VVK] regelt die Vollzugsbestimmungen. Die technischen und grafischen Anforderungen wurden in einer Verordnung des Departements des Innern [VVK-EDI] festgelegt. Diese beinhaltet auch den Standard [eCH-0064] des Vereins eCH, der die technischen Vorgaben regelt.

Grundsätzlich sollen auf der Versichertenkarte Pflichtdaten (Identifikationsdaten Karteninhaber, Angabe zu Kartenherausgeber) gespeichert werden. Das speichern von medizinischen Daten auf der Karte ist möglich, jedoch freiwillig.

Die Einführung der Versichertenkarte wurde in Art 18. [VVK] auf 1. Januar 2009 festgelegt. Diese Frist wurde jedoch auf den 1. Januar 2010[4] verlängert. Ebenso wurde die Einrichtung der online-Abfrage auf den 1. Januar 2010 verlängert.

3.1 Widerstand

Die FMH kritisiert die Einführung der Versichertenkarte. Sie kritisiert Datenschutz sowie Effizienz und Nutzen der Karte und sieht die Patientensicherheit gefährdet z.B. durch fehlerhafte oder unvollständige medizinische Daten auf der Karte. Die FMH schlägt die Nutzung der Versichertenkarte als „digitale Identität“ vor, auf der jedoch keine medizinischen Daten gespeichert werden sollen. Eine HPC (Health Professional Card) soll eingeführt werden, die den elektronischen Austausch von Patientendaten (Berichte, Labordaten, Röntgenbilder usw.) ermöglicht, sofern der Patient einwilligt.[5]

Am 4. Oktober 2007 wurde durch Nationalrat Ruedi Noser eine parlamentarische Initiative „digitale Identität statt Versichertenkarte“ eingereicht. Diese wurde von den Kommissionen beider Räte angenommen. Die Initiative verlangt, die Versichertenkarte nur zur Identikation des Patienten zu benutzen. Medizinische Daten sollen im E-Health System abgelegt werden.[6]

Am 5. September 2007 wurde eine Motion von Ruth Humbel eingereicht, welche die ausser Kraft setzung der VVK verlangte. Dies wurde mit Datenschutzbedenken rund um die medizinischen Daten auf dem Chip begründet. In einer Antwort auf die Motion bestätigt der Bundesrat dass die Versichertenkarte in Zukunft „weniger ein Datenträger als vielmehr ein Zugangsschlüssel zu dezentral verfügbaren Informationen sein sollte“.[7] In Zukunft wird eher auf ein elektronisches Patientendossier fokussiert, statt auf die Speicherung auf dem Chip. Die Motion wurde am 25.09.2009 abgeschrieben, da zu lange hängig.

4 Vgl. BAG Website Versichertenkarte
5 Vgl. Positionspapier der FMH zur „Versichertenkarte mit Gesundheitsdaten“ 18.12.2007
6 Vgl. 07.472 – Parlamentarische Initiative „Digitale Identität statt Versichertenkarte“
7 Antwort des Bundesrats auf die Parlamentarische Initiative „Digitale Identität statt Versichertenkarte“

4 Hersteller

In der Schweiz gibt es zwei Hersteller für **Versichertenkarten**:

- SASIS AG
- Die schweizerische POST

Die **HPC** wird unabhängig davon von der FMH hergestellt.

Für die Benutzung der Versichertenkarte in Spitälern stellt die Firma emineo AG eine SOA (Service oriented architecutre) basierernde **Softwarelösung** her. Für den Einsatz in Arztpraxen bietet Vitodata eine Hard- und Softwarelösung an.

4.1 SASIS AG / VEKA-Center [8]

Die Firma SASIS AG stellt im Auftrag von Santésuisse eine Kunden- und EU-Krankenversicherungskarte her und bietet zusätzlich Abfragedienste an, über die folgende Informationen abgefragt werden können:

- Informationen zum Krankenversicherer
- Basisdaten zum Versicherten
- Deckungsinformationen Grundversicherung
- Deckungsinformationen Zusatzversicherung

Für die spezifische Lösung der SASIS Karte wurden Detailspezifikationen veröffentlicht.

4.2 Die schweizerische Post [9]

Die schweizerische Post stellt im Auftrag der Versicherungsgesellschaft Helsana für diese die Versichertenkarte her. Die Helsana-Gruppe ist der grösste Krankenversicherer der Schweiz.[10]

4.3 FMH – Verbindung der Schweizer Ärztinnen und Ärzte[11]

Die FMH stellt eine der [eCH-0064]–Spezifikation entsprechende HPC her. Diese wurde bereits an über 10'000 FMH Mitglieder ausgestellt.

4.4 Emineo AG[12]

Emineo ist ein Beratungsunternehmen, das sich auf integrierte Lösungen im Gesundheitswesen spezialisiert. Die hergestellte SOA basierte Lösung kann z.B. In SAP integriert werden. Die Software wird gemäss emineo zur Zeit bei 20 Spitälern in der Schweiz eingeführt und soll z.T. bereits im „live“ Einsatz sein.

4.5 Vitodata[13]

Die Firma Vitodata bietet seit Februar 2010 ein Kartenlesegerät für die Integration der Versichertenkarte in ihre Administrationssoftware für Praxen an. (Medizin, Zahnmedizin, Physiotherapie)

8 Website: http://www.veka-center.ch/
9 Website: http://www.post.ch/healthcare
10 Vgl. http://www.helsana.ch
11 Website: http://www.fmh.ch/service/mitgliederausweis_hpc.html
12 Website: http://www.emineo.ch/
13 Website: http://www.vitodata.ch/

5 Auslieferungsstand

5.1 Auslieferung durch Versicherungsgesellschaften

Gemäss dem eHealth Newsletter[14] wurde im Dezember 2009 die Produktion der Karten begonnen. Diese könne mehrere Monate in Anspruch nehmen.
Zum aktuellen Zeitpunkt werden die neuen Versichertenkarten von den Versicherungsgesellschaften ausgeliefert. Die meisten Gesellschaften müssten gemäss Angaben von SASIS[15] die Auslieferung bereits abgeschlossen haben. Der Auslieferungsplan sieht vor, dass beide Kartenherausgeber (SASIS und Post) ihre Auslieferungen bis ende Mai abgeschlossen haben. Es liegt in der Entscheidung der einzelnen Versicherer, wie sie die Auslieferung an die Versicherten durchführen.[16]
Es gibt einzelne, renitente Versicherer, welche die neue Versichertenkarte nicht einführen möchten.[17] Dies stellt gemäss BAG ein aufsichtsrechtliches Problem dar.

14 Vgl. http://www.e-health-suisse.ch
15 Vgl. „Produktionsforschritt Versichertenkarte 2010 – Stand 19. März 2010“ http://www.veka-center.ch
16 Vgl. Rückmeldung von Santésuisse zum Brief von H+ vom 21.12.2009, 7.1.2010
17 Vgl. Sitzungsprotokoll „Weiteres Vorgehen Versichertenkarte – Runder Tisch“ 3.12.2009

6 Technischer Entwicklungsstand

Die aktuell ausgelieferte Karte enthält Pflichtdaten (Identifikationsdaten Karteninhaber, Angabe zu Kartenherausgeber) sowie vorbereitete Ordner (DF) und Dateien (EF) für die Speicherung von medizinischen Notfalldaten und für kantonale Modellversuche. Sicherheitsmechanismen und Zertifikate für die Authentifizierung mit HPC Zertifikaten sowie das PIN Management sind bereits implementiert.
Es gibt zur Zeit allerdings Kompatibilitätsprobleme zischen den beiden Implementierungen von Post und SASIS, da diese unterschiedliche Zertifikate benutzen.[18] Die Abfrage der administrativen Daten ist davon jedoch nicht betroffen, da diese ohne Zertifikate möglich ist.

6.1 Health Professional Card (HPC)

Rund ein Drittel der Mediziner mit Patientenkontakt (> 10'000 gemäss FMH) haben bereits eine HPC erhalten. Diese HPC ist jedoch nicht mit der Versichertenkarte der SASIS AG kompatibel, da sich die SASIS gemäss der FMH nicht an den vom Bund herausgegebenen Standard gehalten habe[19]. Es wird nun versucht, zwei Zertifikate auf der HPC aufzubringen. Die technische Machbarkeit ist noch nicht nachgewiesen.

6.2 Medizinische Daten auf der Karte

Zur Zeit gibt es keine Möglichkeit, medizinische Daten von der Versichertenkarte zu lesen oder auf diese zu schreiben. Dies wäre nur mit gültigem HPC Zertifikat möglich und wird momentan durch die oben genannten Kompatibilitätsprobleme verhindert.
Die FMH kritisiert zudem die statische Zuteilung des Speicherplatzes auf der Karte. Dieser sei für die medizinischen Daten z.T. zu klein und entspreche nicht der Spezifikation.

6.3 „Online-Verfahren“ - Abfragedienste

Die SASIS AG bietet Abfragedienste an, die mittels Identifikation durch die Versichertenkarte durchgeführt werden können. Die angebotenen Informationen werden im Abschnitt 4.1 genannt.

6.4 Kantonale Modellversuche

Auch bei den für die kantonalen Modellversuche benötigten Zertifikate treten zur Zeit noch Kompatibilitätsprobleme mit den Zertifikaten auf. Gemäss BAG sollen die Modellversuche ohne Zertifikate durchgeführt werden.

6.5 Infrastruktur bei Leistungserbringern

Gemäss einer Kosten/Nutzen Analyse der Firma Debold & Lux wird für Leistungserbringer mit Investitionskosten von durchschnittlich 2'100 Franken gerechnet, damit persönliche (medizinische) Daten bei Versicherten auf die Karte geschrieben werden können. Die Kosten, welche durch die Aufnahme der Daten im Rahmen von medizinischen Konsultationen entstehen, werden dagegen von der obligatorischen Krankenpflegeversicherung bezahlt.[20]

6.5.1 Eingesetzte Lesegeräte

Für den Einsatz mit der HPC wurden von der FMH folgende beiden Lesegeräte geprüft:

- Cherry USB Chipkartenleser ST-1044UB
- Cherry PCMCIA Chipkartenleser für Notebooks SR-4044

Grundsätzlich sollten Versichertenkarte und HPC mit den meisten handelsüblichen Kartenlesegeräten funktionieren.[21]

6.5.2 Eingesetzte Software

Momentan steht den Leistungserbringern noch keine Software zur Verfügung, um medizinische Daten auf die Versichertenkarte zu schreiben. Gemäss FMH gibt es erst Testversionen.
Auch die von der emineo AG angebotene Softwarelösung unterstützt zur Zeit noch keine medizinischen Daten. Das Auslesen der administrativen Daten d.h. die Nutzung der Versichertenkarte zur Identifikation funktioniert hingegen.
Auch die Softwarelösung der Firma Vitodata unterstützt zur Zeit die Bearbeitung der medizinischen Daten nicht.

18 Vgl. Sitzungsprotokoll „Weiteres Vorgehen Versichertenkarte – Runder Tisch“ 3.12.2009
19 Vgl. „CH/Neue Versichertenkarte: Ärzte kritisieren Informatikprobleme“ Handelszeitung.ch - 18.2.2010
20 Vgl. Kosten/Nutzen Analyse Debold & Lux, 08. 6. 2006
21 Vgl. FMH Website HPC

7 Schlussfolgerungen

Die Einführung der Versichertenkarte in der Schweiz ist nun definitiv im Gange. Die Kartenherstellung soll Ende Mai 2010 abgeschlossen sein. Wann die Karten ausgeliefert werden, entscheiden die einzelnen Versicherungsgesellschaften.

Die Nutzung der Karte wird aber vermutlich eher in Richtung „digitale Identität“ gehen, d.h. der Speicherung von medizinischen Daten auf der Chipkarte wird weniger Gewicht beigemessen. Es ist zur Zeit noch gar nicht klar, ob das freiwillige Speichern von medizinischen Daten auf dem Speicherchip überhaupt möglich sein wird, da Kompatibilitätsprobleme zwischen den Kartenherstellern existieren und auch noch keine geeignete Software für diesen Zweck erhältlich ist. Sowohl FMH als auch Spitäler haben sich kritisch gegenüber der Speicherung von medizinischen Daten auf der Chipkarte geäussert. Sie werden versuchen, diese Option wieder aus dem Gesetz zu entfernen.[22]

Für die Nutzung der Versichertenkarte zur elektronischen Identifizierung von Patienten mittels AHV Nummer/Kartennummer wird zur Zeit noch auf Praxis- und Spitalebene Software eingeführt.

Gemäss Bund soll die Einführung der Versichertenkarte den Weg für das elektronische Patientendossier ebnen (geplante Einführung: 2015).

22 Vgl. Sitzungsprotokoll „Weiteres Vorgehen Versichertenkarte – Runder Tisch“ 3.12.2009

8 Anhang

8.1 Literaturverzeichnis

Zulauf, Carla, BAG (3.12.2009) Sitzungsprotokoll Weiteres Vorgehen Versichertenkarte – Runder Tisch. Online unter URL: http://www.hplus.ch/de/ehealth_statistik/ehealth/versichertenkarte/ Stand: 25.3.2010

OFAC (26.9.2008) Gegensätzliche Debatte über die Zukunft der Versicherten- und Gesundheitskarte. Online unter URL: http://www.ofac.ch/de/media/index.htm Stand: 26.3.2010

BAG Website Versichertenkarte. Online unter URL: http://www.bag.admin.ch/themen/krankenversicherung/07060/index.html?lang=de Stand: 26.3.2010

Anon. 07.472 – Parlamentarische Initiative „Digitale Identität statt Versichertenkarte“. Online unter URL: http://www.parlament.ch/D/Suche/Seiten/geschaefte.aspx?gesch_id=20070472 Stand: 26.3.2010

Allemann, Dominic SASIS AG (19. 3.2010) Produktionsfortschritt Versichertenkarte 2010 – Stand 19. März 2010. Online unter URL: http://www.veka-center.ch/ Stand: 25. März 2010

Kaufmann, Stefan Santésuisse (7.1.2010) Rückmeldung von Santésuisse zum Brief von H+ vom 21.12.2009 Online unter URL: http://www.hplus.ch/de/ehealth_statistik/ehealth/versichertenkarte/ Stand: 25. März 2010

Anon. (18.2.2010) CH/Neue Versichertenkarte: Ärzte kritisieren Informatikprobleme. Online unter URL: http://www.handelszeitung.ch/artikel/Unternehmen-AWP_CH_Neue-Versichertenkarte-aerzte-kritisieren-Informatikprobleme_685580.html Stand: 24. März 2010

Debold & Lux (8.6.2010) Kosten/Nutzen Analyse - Die Versichertenkarte und der Aufbau einer Telematikinfrastruktur. Online unter URL: http://www.bag.admin.ch/themen/krankenversicherung/07060/index.html?lang=de Stand: 25. März 2010

Anon. Website FMH zur HPC Online unter: http://www.fmh.ch/service/mitgliederausweis_hpc/benutzung.html Stand: 25. März 2010

8.2 Gesetze und Standards

KVG, Bundesgesetz vom 18. März 1994 über die Krankenversicherung (KVG). Online unter URL: http://www.admin.ch/ch/d/sr/c832_10.html Stand: 26. März 2010

VVK, Verordnung vom 14. Februar 2007 über die Versichertenkarte für die obligatorische Krankenpflegeversicherung (VVK). Online unter URL: http://www.admin.ch/ch/d/sr/c832_105.html Stand: 26. März 2010

VVK-EDI, Verordnung vom 20. März 2008 über die technischen und grafischen Anforderungen an die Versichertenkarte für die obligatorische Krankenpflegeversicherung (VVK-EDI) Online unter URL: http://www.bag.admin.ch/themen/krankenversicherung/07060/index.html?lang=de Stand: 26. März 2010

eCH-0064 (4.2.2008) Spezifikationen für das System Versichertenkarte. Online unter URL: http://www.ech.ch/vechweb/page?p=dossier&documentNumber=eCH-0064&documentVersion=1.00 Stand: 25. März. 2010

Elektronische Versichertenkarte

Beschaffung

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-03-08
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Evaluation von Kartenlesegeräten.

1.2 Zweck

Dieses Dokument beschreibt die Evaluation von verschiedenen Kartenlesegeräten für die Verwendung mit dem Demonstrator. Es werden Anforderungen an Karte und Gerät definiert und verschiedene Produkte verglichen.

1.3 Gültigkeit

Dieses Dokument dient als Grundlage für die ganze Arbeit und hat deshalb Gültigkeit über die gesamte Dauer des Projekts.

2 Anforderungen

Die Anforderungen an Chipkarte und Lesegerät sind im Standard [eCH-0064] definiert und werden im Dokument „Detailspezifikationen Version 1.3 vom 6.1.2010 zur Versichertenkarte nach VVK 832.105“ (www.sasis.ch) erweitert.

2.1 Anforderungen an die Versichertenkarte

Typ	*Anforderung*	*Typ*
Physikalische Anforderungen	• ISO/IEC 7816-1 • ISO/IEC 7816-2	zwingend
Kommunikations-Anforderungen	Übertragungsverfahren gemäss ISO/IEC 7816-3: • Übertragungsprotokoll T=1 mit Chaining • Keine Knotenadressierung • S-Block ABORT [PCB] nicht verwenden (empfohlen) • Grösse Informationsfelder IFSC >= 128 Byte, IFSD = 254 Byte • Protokoll-Parameter-Auswahl (PPS) mit Unterstützung des aushandelbaren Modus, sonst Teiler (CRFC) auf die Werte 372 oder 512, damit mindestens 38kbps oder höher gewährleistet werden können • ATR-Kodierung übereinstimmend mit ISO/IEC 7816-3	zwingend
Betriebssystem	Unterstützung minimaler Befehlssatz gemäss ISO/IEC 7816-4. Befehlssatz ist in [eCH-0064] definiert.	zwingend
	Unterstützung folgender kryptologische Operationen: • GENERATE ASSYMMETRIC KEY PAIR • '7F49' INTERINDUSTRY TEMPLATE • PERFORM SECURITY OPERATION	zwingend
Speicher	Der EEPROM Speicherbereich der Chipkarte muss über mindestens 32KByte verfügen	zwingend

Initialisierung / Personalisierung	• Minimaler Befehlssatz gemäss ISO/IEC 7816-9 muss unterstützt werden:
	  • CREATE FILE
	  • DELETE FILE
	  • DEACTIVATE FILE
	  • ACTIVATE FILE
	  • TERMINATE DF
	  • TERMINATE EF
	  • TERMINATE CARD USAGE
		zwingend

Authentisierung	• Unterstützung von CVC Zertifikaten gemäss ISO/IEC 7816-6/8 • Unterstützung von RSA Schlüssellänge 1024 Bit • Hashfunktion SHA1 (160 Bit Hashwert)	zwingend
Dateisystem	Dateiverwaltung muss dem Standard ISO/IEC 7816-4 genügen	zwingend
PIN-Management	Unterstützung Befehlssatz gemäss ISO/IEC 7816-4	zwingend

Table 1: Anforderungen an die Versichertenkarte

2.2 Anforderungen an den elektronischen Leistungserbringernachweis (HPC)

Typ	*Anforderung*	*Typ*
Physikalische Anforderungen	• ISO/IEC 7816-1,2,3,4,5,6,8,9 • ID-1-Format (nach ISO 7810) (kann bei Bedarf herausgebrochen werden in das ID-000 Format)	zwingend

Table 2: Anforderungen an den elektronischen Leistungserbringernachweis (HPC)

2.3 Anforderungen an das Lesegerät (Terminal)

Typ	*Anforderung*	*Typ*
Physikalische Anforderungen	USB-Anschluss	empfohlen
Kommunikations-Anforderungen	Unterstützung Übertragungsprotokoll T=1	zwingend
Lese / Schreibfrequenz	Kompatibel mit Versichertenkarte (Programmierfrequenz)	zwingend
Software-Ansteuerbarkeit	Ansteuerbar über Java API	zwingend
PIN-Eingabe direkt am Gerät	Eingabe PIN-Code direkt am Lesegerät	empfohlen

Table 3: Anforderungen das Lesegerät (Terminal)

3 Beschaffung

3.1 Versichertenkarte

Die schweizerischen Versichertenkarten werden von der Post und der Firma SASIS AG (www.sasis.ch) vertrieben. Die Firma SASIS AG stellt uns kostenlos eine reale schweizerische Gesundheitskarte für Tests zur Verfügung.

3.2 Leistungserbringernachweis (HPC)

Es existiert ein Leistungserbringernachweis in Form einer Health Professional Card (HPC). Dieser wird von der FMH, der Verbindung Schweizer Ärzte und Ärztinnen vertrieben.

3.3 Lesegerät

Ein geeignetes Lesegerät für die Chipkarte muss beschafft werden. Im folgenden Kapitel werden Geräte verglichen.

4 Vergleich Lesegeräte

4.1 Vergleichsmatrix Kartenlesegeräte

Vergleichsmatrix der Kartenlesegeräte für den Einsatz mit der elektronischen Gesundheitskarte.

Vergleichsmatrix Kartenlesegeräte

Für den Einsatz mit der elektronischen Gesundheitskarte

		Cherry SmartCard Reader ST-1044UB		SCM Microsystems Chipdrive Micro Pro S230201		Cherry SmartTerminal ST-2000UC-R		SCM Microsystems Pinpad Pro SPR332		SCM Microsystems eHealth 100		SCM Microsystems eHealth 200BCS		SCM Microsystems Chipdrive Desktop Pro S230203	
Funtionale Kriterien	*Gewichtung*	*Bewertung*	*Gewichtet*	*Bewertung*	*Gewichtet*	*Bewertung*	*Gewichtet*	*Bewertung*	*Gewichtet*	*Bewertung*	*Gewichtet*	*Bewertung*	*Gewichtet*	*Bewertung*	*Gewichtet*
Übertragungsprotokolle (mindestens T=1)	6	10	60	6	36	10	60	6	36	6	36	6	36	6	36
Verschiedene Programmierfrequenzen (4,8mHz bis 8mHz)	3	6	18	8	24	6	18	8	24	8	24	8	24	8	24
Unterstützung versch. Betriebssysteme	4	6	24	10	40	6	24	10	40	10	40	10	40	10	40
USB-Anschluss	6	10	60	10	60	10	60	10	60	10	60	10	60	10	60
Direkte PIN-Eingabe am Lesegerät	4	0	0	0	0	8	32	8	32	10	40	10	40	0	0
Unterstützung SIM-Kartenformat	3	0	0	8	24	0	0	8	24	8	24	8	24	8	24
Kompatibilität JAVA API (PC/SC Treiber)	7	8	56	8	56	8	56	8	56	8	56	8	56	8	56
Separater Karteneinschub für HPC	3	0	0	0	0	0	0	0	0	10	30	10	30	0	0
Funktionale Kriterien gewichtet:			**218**		**240**		**250**		**272**		**310**		**310**		**240**
Organisatorische und kommerzielle Kriterien	*Gewichtung*	*Werte*	*Gewichtet*	*Werte*	*Gewichtet*	*Werte*	*Gewichtet*	*Werte*	*Gewichtet*	*Werte*	*Gewichtet*	*Werte*	*Gewichtet*	*Werte*	*Gewichtet*
Lieferfrist	8	2-3 Tage	64	2-3 Tage	64	2-3 Tage	64	3-4 Tage	40	k.A.	0	k.A.	0	10 Tage	16
Garantie	6	2 Jahre	12	k.A	0	2 Jahre	12	k.A	0	k.A	0	k.A	0	k.A	0
Mengenrabatt (evtl. wichtig bei späterer Beschaffung Schulungsgeräte)	4	> 5 Stück	4	> 5 Stück	4	> 5 Stück	4	nein	0	k.A	0	k.A	0	> 5 Stück	4
Preis		47.00 Fr.		79.00 Fr.		79.00 Fr.		61.50 €		278.00 €		345.00 €		99.00 Fr.	
Preis CHF	6	47.00 Fr.	63.83 Fr.	79.00 Fr.	37.97 Fr.	79.00 Fr.	37.97 Fr.	91.50 Fr.	32.79 Fr.	417.00 Fr.	7.19 Fr.	517.00 Fr.	5.80 Fr.	99.00 Fr.	30.30 Fr.
Bestellbar bei		distrelec.ch		distrelec.ch		distrelec.ch		chipdrive.de		scmmicro.com		e-medicalcard.de		distrelec.ch	
Bemerkungen		Stabil, da hohes Gewicht.								Speziell für deutsche eGK entwickelt. Firmware unter GPL. Veröffentlicht.				Lieferfrist ungewiss.	
Organisatorische und kommerzielle Kriterien gewichtet:			**143.83**		**105.97**		**117.97**		**72.79**		**7.19**		**5.80**		**50.30**
Total gewichtet:			**362**		**346**		**368**		**345**		**317**		**316**		**290**
Rang:			**2**		**3**		**1**		**4**		**5**		**6**		**7**

Illustration 1: Vergleichsmatrix Kartenlesegeräte

5 Investitionsantrag

5.1 Antragsteller

Antragsteller	
Name	Michael Hofmann mhofmann@hsr.ch Mark Lowe mlowe@hsr.ch
Datum	18.März 2010
Betreuender Dozent	Axel Doering adoering@hsr.ch
Abteilung	Informatik
Arbeit	Bachelorarbeit
Grund	Ein mit der schweizerischen elektronischen Versichertenkarte kompatibles Lesegerät wird für die Entwicklung einer Demonstrationsapplikation benötigt.

Table 4: Investitionsantrag Antragssteller

5.2 Beschaffungsvorschlag

Beschaffungsvorschlag	
Hersteller	Cherry
Gerät	Cherry Smart Terminal
	ST-2000UC-R
Lieferant	distrelec.ch
Kaufpreis	CHF 73.42 + CHF 12.40 Versandkosten + MWST **= CHF 92.34**
Liefertermin	nächster Arbeitstag
Zahlungsbedingungen	Zahlung per Rechnung nach Warenerhalt
Begründung	Dieses Kartenleseterminal bietet zusätzliche Sicherheit dank PIN-Eingabe direkt am Lesegerät. Für die Demonstrationsapplikation kann damit eine realitätsgetreuere Demo erzeugt werden. Es erfüllt alle Anforderungen, gleich wie die beiden Alternativgeräte.

Table 5: Investitionsantrag Beschaffungsvorschlag

5.3 Alternativen

Alternative 1	
Hersteller	Cherry
Gerät	Cherry SmartCard Reader
	ST-1044UB
Lieferant	distrelec.ch
Kaufpreis	CHF 43.68 + CHF 12.40 Versandkosten + MWST **= CHF 60.34**
Liefertermin	nächster Arbeitstag
Zahlungsbedingungen	Zahlung per Rechnung nach Warenerhalt
Begründung	Günstigstes Modell in der Auswahl. Erfüllt alle Anforderungen, hat allerdings keine direkte Pin-Eingabe.

Table 6: Investitionsantrag Alternative 1

Alternative 2	
Hersteller	SCM Microsystems
Gerät	Chipdrive Micro Pro
Modellnummer	S230201
Lieferant	distrelec.ch
Kaufpreis	CHF 73.42 + CHF 12.40 Versandkosten + MWST **= CHF 92.34**
Liefertermin	nächster Arbeitstag
Zahlungsbedingungen	Zahlung per Rechnung nach Warenerhalt
Begründung	Erfüllt alle Anforderungen, hat aber keine direkte PIN-Eingabe.

Table 7: Investitionsantrag Alternative 2

6 Anhang

6.1 Abbildungsverzeichnis

6.2 Tabellenverzeichnis

6.3 Literaturverzeichnis

eCH-0064: Adrian Schmid, Jürg Burri, Willy Müller, Peter Stadlin, Martin Stingelin, , 2008

Elektronische Versichertenkarte

Anforderungen

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-03-19
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Spezifikation der Anforderungen des Demonstrators.

1.2 Zweck

Zweck ist die Dokumentation der Anforderungen.

1.3 Gültigkeit

Dieses Dokument dient als Grundlage für den Demonstrator und hat deshalb Gültigkeit über die gesamte Dauer der Arbeit am Demonstrator. Änderungen werden laufend ergänzt und sind durch die Versionskontrolle ersichtlich.

2 Allgemeine Beschreibung

Es soll eine Experimentierumgebung für die elektronische Gesundheitskarte entwickelt werden. Die Experimentierumgebung soll als Demonstrator für die Lehre eingesetzt werden.

2.1 Funktionen

Es sollen im Minimum die Pflichtfelder einer spezifikationsgemässen Versichertenkarte ausgelesen werden. Diese Felder sollen über einen Webbrowser angezeigt werden.

2.2 Charakteristiken

Zum Auslesen der elektronischen Gesundheitskarte soll ein SmartCard Reader verwendet werden. Dieser soll direkt mit dem Computer verbunden sein, welcher die entsprechenden Felder auf der elektronischen Gesundheitskarte ausliest.

2.2.1 Optionale Funktionen

- Schreiboperationen auf der elektronischen Gesundheitskarte
- Komplexe Transaktionen auf der elektronischen Gesundheitskarte

2.3 Einschränkungen

Es müssen nur spezifikationsgemässe Versichertenkarten ausgelesen werden.

2.4 Abhängigkeiten

Da die Felder der Versichertenkarte über einen Webbrowser angezeigt werden sollen, soll der Demonstrator möglichst Plattformunabhängig implementiert werden.

2.5 Benutzer

Die Benutzer des Demonstrators sind Studenten ab dem 5. Semester, die an einer Vertiefungsveranstaltung „Medizinische Informationssysteme“ teilnehmen.

3 Funktionale Anforderungen

3.1 Allgemein

Ziel ist es, ein System zu entwickeln, mit welchem bestimmte Daten einer elektronischen Gesundheitskarte ausgelesen und mit einer grafischen Benutzerschnittstelle angezeigt werden können. Als elektronische Gesundheitskarte soll dabei die schweizerische Versichertenkarte nach Spezifikation eCH-0064 eingesetzt werden.

3.2 Daten

Die auszulesenden Daten befinden sich auf der elektronischen Gesundheitskarte. Dabei gibt es verschiedene Kategorien von Daten. Es können dabei nur Pflichtfelder ohne entsprechende Authorisierung ausgelesen werden. Das lesen und/oder schreiben weiterer Daten ist abhängig davon, ob eine entsprechende Authorisierung dazu vorliegt. Es sollen alle Pflichtfelder ausgelesen werden. Das lesen der Nutzdaten soll ebenfalls implementiert werden, ist aber abhängig von der Beschaffung entsprechender Karten für Daten und Authentifizierung (HPC und Versichertenkarte der Post). Falls die benötigten Karten nicht oder nicht rechtzeitig beschafft werden können, sollen für diese Karten je ein Objekt erstellt werden, welches diese Karten repräsentiert (virtuelle Karten). Dadurch soll ermöglicht werden, das Potenzial der Versichertenkarten auch ohne die benötigten Karten zu demonstrieren.

3.2.1 Internal

Internal bezeichnet interne Daten, auf die nicht direkt zugegriffen werden kann. Diese Daten werden ausschliesslich von der elektronischen Gesundheitskarte selber angelegt und verwendet. Ein Auslesen oder Ändern der Daten ist nicht möglich.

3.2.2 Working

Working bezeichnet Daten, mit denen gearbeitet werden kann. Solche Daten können gelesen und/oder geschrieben werden.

3.2.2.1 Pflichtfelder

Unter Pflichtfelder versteht man alle working Daten, die sich direkt unterhalb des Masterfiles befinden. Diese können auch alle direkt gelesen und/oder geschrieben werden. Eine Authentifizierung ist nicht nötig.

3.2.2.2 Nutzdaten

Unter Nutzdaten versteht man alle working Daten, die sich in einem Dedicated File (Verzeichnisdatei) befinden. Diese Daten gehören alle zu einer bestimmten Anwendung. Alle Daten einer Anwendung werden jeweils in einem eigenen Dedicated File abgelegt. Einige Daten dieser Anwendungen können nur mit vorangehender Authentisierung und Autorisierung der entsprechenden Leistungserbringergruppe gelesen und geschrieben werden.

3.3 Zugriffskontrolle & Authentifizierung

Benutzerauthentifizierung und Zugriffskontrolle sind nicht nötig. Es müssen entsprechend auch keine Benutzergruppen auf Applikationsebene implementiert werden.
Für den Zugriff auf bestimmte Daten der elektronischen Gesundheitskarte müssen die im Standard eCH-0064 definierten Sicherheitsanforderungen eingehalten werden. Dabei kann eine Authentifizierung des Karteninhabers nötig sein, sowie eine Leistungserbringer-Authentifizierung mittels Health Professional Card.

3.3.1 Benutzer

Es gibt keine spezifischen Benutzer und auch keine Benutzergruppen. Alle Benutzer können sämtliche Funktionen des Demonstrators auf gleiche weise verwenden.

3.4 Benutzerschnittstelle

Die Benutzer müssen über einen Webbrowser auf die Benutzerschnittstelle zugreifen können. Es muss eine grafische Benutzerschnittstelle implementiert werden. Die Abläufe bei der Verwendung der elektronischen Gesundheitskarte sollten möglichst klar ersichtlich sein.

4 Nichtfunktionale Anforderungen

4.1 Leistungsanforderungen

Die Benutzerschnittstelle zum Demonstrator muss in einem Webbrowser lauffähig sein.

4.2 Mengenanforderungen

Da der Demonstrator zu Demonstrationszwecken in der Lehre eingesetzt wird, ist er auf einen Betrieb mit bis zu 30 Benutzer gleichzeitig zu optimieren. Der Demonstrator muss aber von beliebig vielen Benutzern gleichzeitig verwendet werden können, begrenzt durch die Systemressourcen.

4.3 Schnittstellenanforderungen

4.3.1 Benutzerschnittstelle

Der Benutzer muss auf den Demonstrator über einen Webbrowser zugreifen können.

4.3.2 Softwareschnittstelle

Es sollen klare Schnittstellen definiert werden um eine Fortsetzungsarbeit zu erleichtern.

4.4 Randbedingungen

4.4.1 Softwareeinschränkungen

Der Demonstrator soll möglichst Betriebssystemunabhängig implementiert werden. Als Minimalanforderung gilt die Unterstützung folgender Betriebssysteme:

- Windows 7
- Windows XP
- Linux
- Mac OS X

Die Betriebssystemunabhängigkeit ist jedoch auch von den Treibern für das Kartenlesegerät abhängig.

4.4.2 Hardwareeinschränkungen

Der Demonstrator soll möglichst Hardwareunabhängig implementiert werden. Als Minimalanforderung gilt die Unterstützung folgender Plattformen:

- x86
- AMD64

Die Plattformunabhängigkeit ist jedoch auch von den Treibern für das Kartenlesegerät abhängig.

4.4.3 Peripherieeinschränkungen

Einschränkungen bei den Peripheriegeräten ergeben sich durch die Spezifikation der elektronischen Gesundheitskarte, definiert im Standard eCH-0064. Sowohl die eingesetzten elektronischen Gesundheitskarten als auch die entsprechenden Kartenleser müssen sich nach dem Standard ISO/IEC 7816 richten.

4.5 Qualitätsmerkmale

4.5.1 Funktionalität

Sämtliche Funktionalitäten sind entsprechend den Anforderungen zu implementieren und zu Testen. Zur Gewährleistung der Interoperabilität soll auf plattformspezifische Eigenheiten bei Möglichkeit verzichtet werden.

4.5.2 Zuverlässigkeit

Der Demonstrator muss absolut korrekt Arbeiten. Die Fehlertoleranz besteht soweit, dass es zu Kommunikationsproblemen mit dem Lesegerät oder der elektronischen Gesundheitskarte kommen kann. Probleme dieser Art müssen erkannt und der Benutzer mit entsprechenden Fehlermeldungen darüber informiert werden.

4.5.3 Benutzbarkeit

Die Benutzbarkeit hat sich nach dem Umstand zu richten, dass der Demonstrator zu Demonstrationszwecken in der Lehre eingesetzt wird. Der Fokus bei der Entwicklung der Benutzeroberfläche soll bei deren Verständlichkeit und nicht bei deren Effizienz liegen. Die Benutzeroberfläche muss so gestaltet werden, dass sie mit möglichst wenig Vorkenntnissen bedienbar ist. Die Bedienung muss möglichst intuitiv erfolgen. Ausserdem sollen sämtliche Abläufe durch die Benutzeroberfläche möglichst klar ersichtlich werden.

4.5.4 Sicherheit

Sicherheitsanforderungen auf Seiten des Demonstrators sind nicht gefordert. Eine Authentifizierung oder Zugriffskontrolle für die Benutzer ist nicht zu implementieren. Sicherheitsanforderungen bezüglich der Verwendung der elektronischen Gesundheitskarte sind definiert im Standard eCH-0064 und müssen entsprechend eingehalten werden.

4.5.5 Effizienz

Es wird kein bestimmtes minimales oder maximales Zeitverhalten gefordert. Das Verbrauchsverhalten sollte dasjenige eines Standard Desktop-Computers (ca. 2 GHz CPU, 2 GB RAM, 120 GB HD) nicht übersteigen.

4.5.6 Analysierbarkeit

Es müssen keine Logging-Informationen erstellt und persistiert werden.

4.5.7 Änderbarkeit

Das System ist so zu gestalten, dass es in einer Fortsetzungsarbeit problemlos erweitert, angepasst und modifiziert werden kann.

4.5.8 Prüfbarkeit

Isolierte Codeteile sind mit Unit-Tests zu prüfen. Sämtliche Funktionalitäten werden in Systemtests geprüft.

4.6 Weitere Anforderungen

4.6.1 Konfigurierbarkeit

Die Systemparameter sollen bei Möglichkeit ausserhalb des Programmcodes definiert werden um sie auch nach der Installation ändern zu können.

4.6.2 Betrieb

Die Stabilität im Betrieb darf nur beeinträchtigt werden durch:

- Externe Peripheriegeräte
- Systemressourcen des Host-Systems

4.6.3 Inbetriebnahme & Installation

Der Installationsvorgang muss klar und verständlich Dokumentiert werden.

4.6.4 Projektübergabe

Der Demonstrator ist sowohl in Source Code, als auch in kompilierter Form zu übergeben.

4.6.5 Dokumentation

Es sind eine Systemdokumentation und ein Benutzerhandbuch zu erstellen.

5 Anhang

5.1 Abbildungsverzeichnis

5.2 Tabellenverzeichnis

5.3 Literaturverzeichnis

Elektronische Versichertenkarte

Analyse

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-03-22
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Dokumentation der Analyse des Demonstrators.

1.2 Zweck

Zweck ist die Dokumentation der Analyse. Es sollen der grobe Programmaufbau, Abläufe sowie Abmachungen festgelegt werden.

1.3 Gültigkeit

Dieses Dokument dient als Grundlage für den Demonstrator und hat deshalb Gültigkeit über die gesamte Dauer der Arbeit am Demonstrator. Änderungen werden laufend ergänzt und sind durch die Versionskontrolle ersichtlich.

2 Use Cases

2.1 Use Case Model

2.1.1 Use Cases aus pädagogischer Perspektive

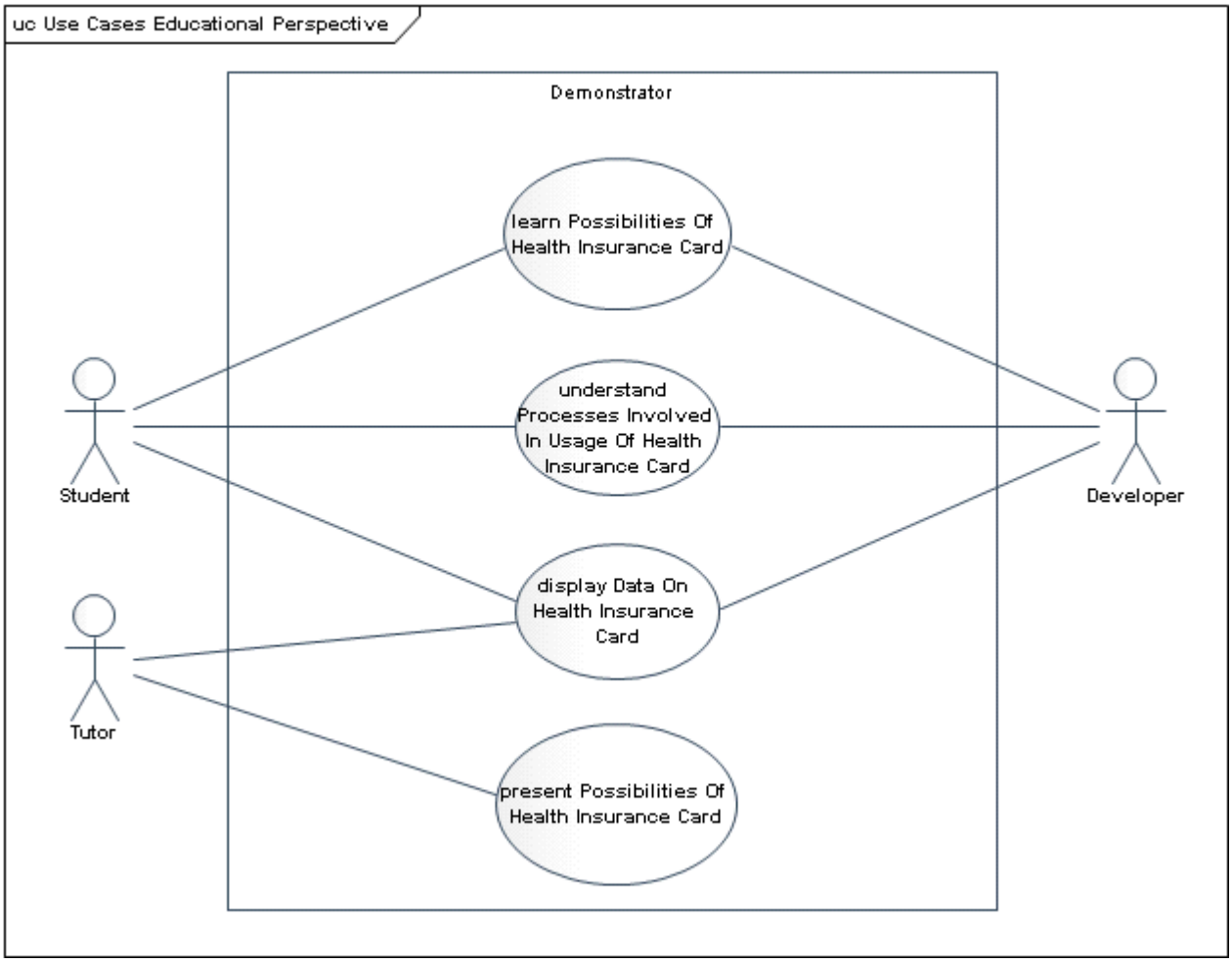


Illustration 1: Use Cases aus pädagogischer Perspektive

2.1.2 Use Cases aus technischer Perspektive

Kurz

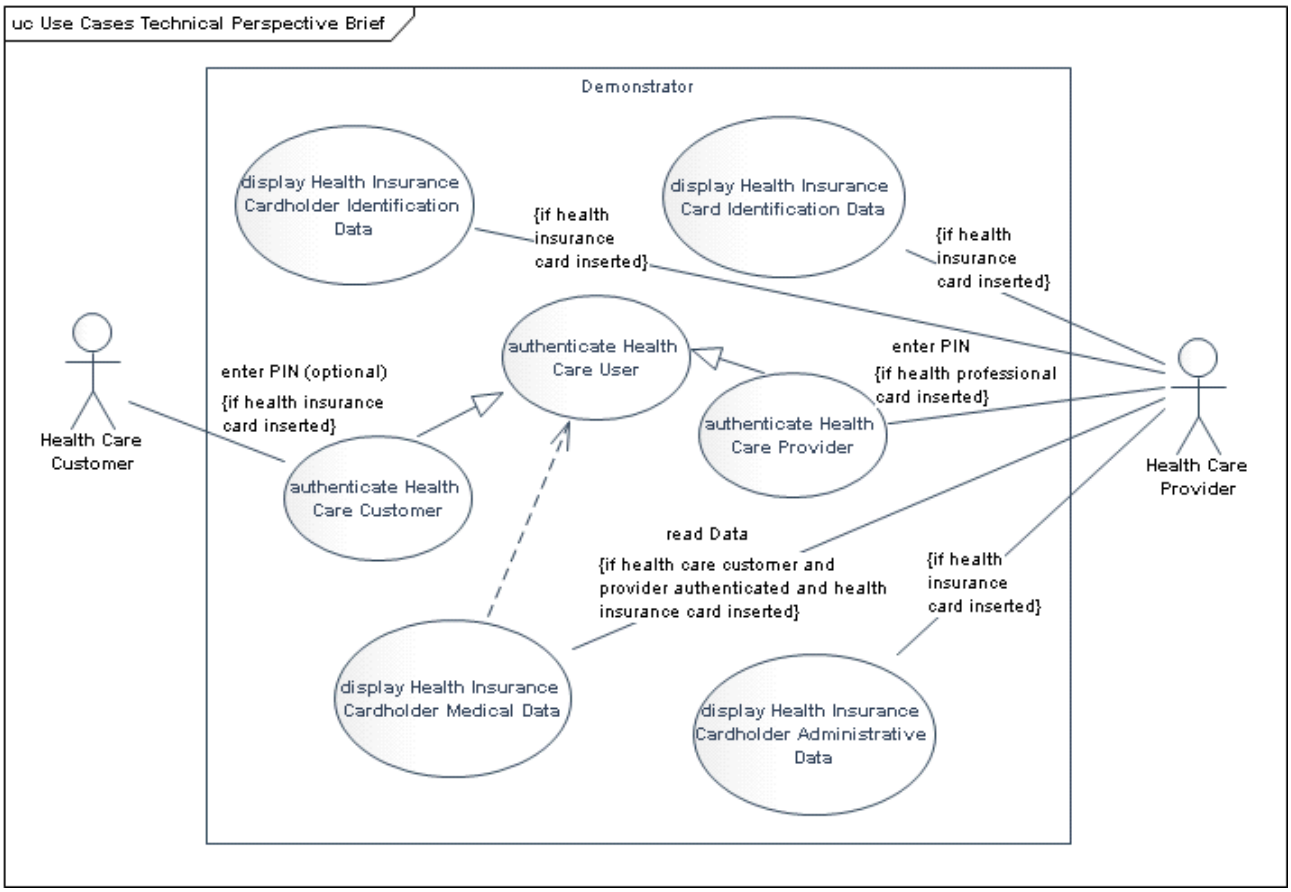


Illustration 2: Use Cases aus technischer Perspektive Kurz

Detailliert

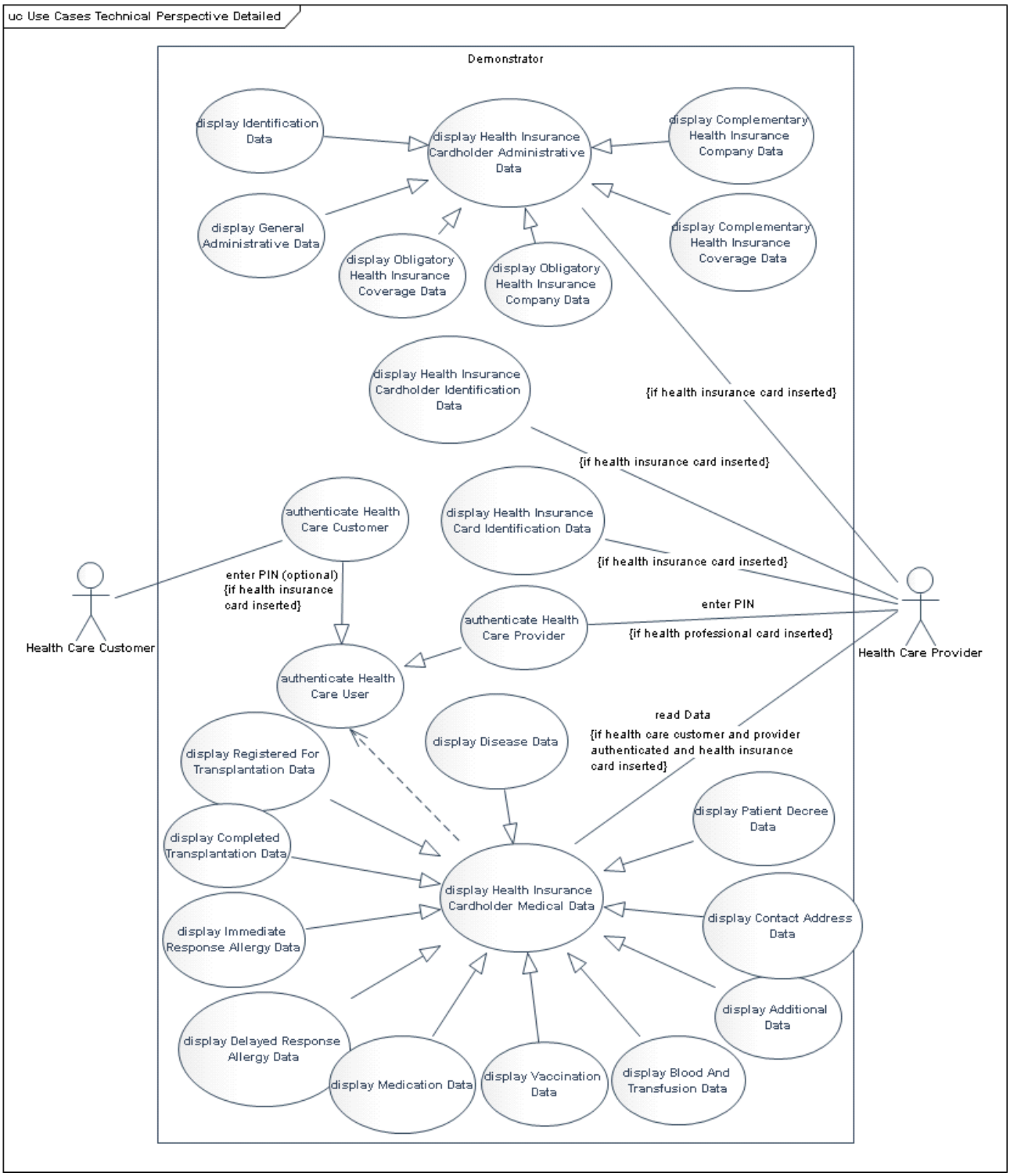


Illustration 3: Use Cases aus technischer Perspektive Detailliert

2.2 Aktoren

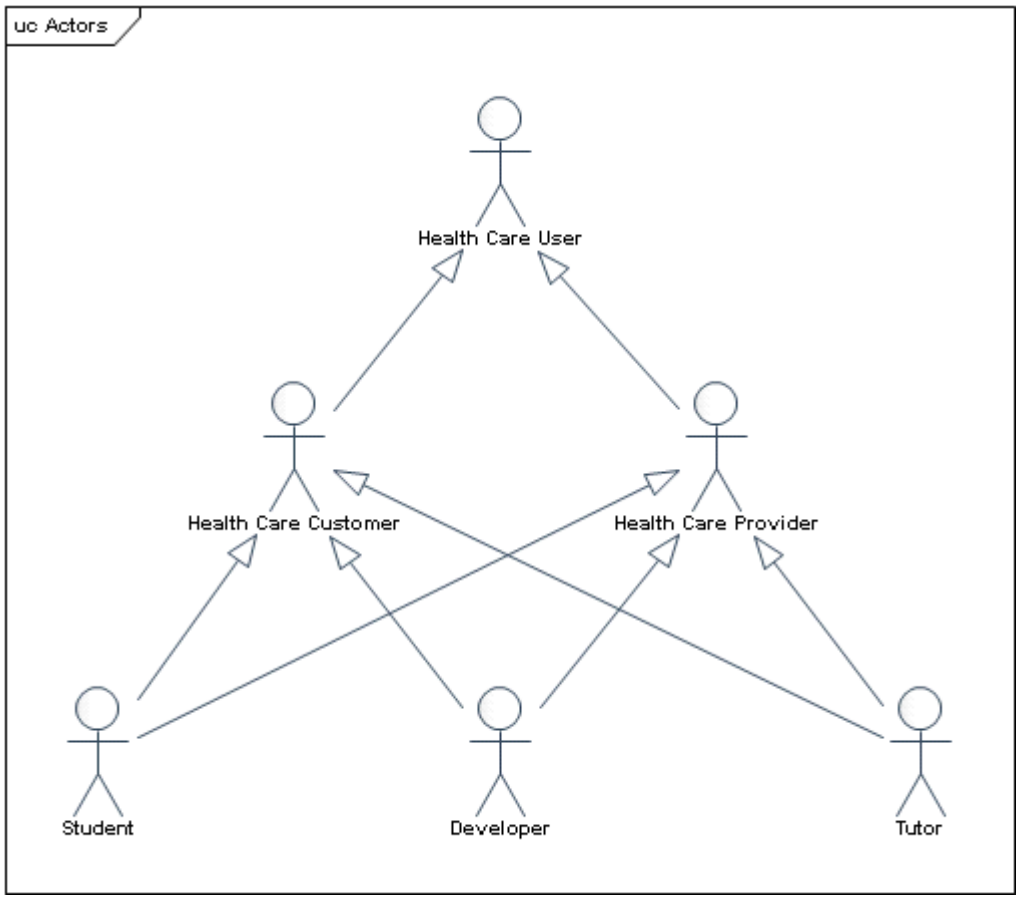


Illustration 4: Aktoren

2.2.1 Health Care User

AC-1	*Health Care User*
Beschreibung:	Health Care User sind alle Personen, welche den Demonstrator verwenden. Grundsätzlich werden zwei Gruppen von Benutzern unterschieden: Health Care Customer und Health Care Provider. Diese Benutzergruppen sind als Rollen zu verstehen, welche von den verschiedenen Endbenutzer-Gruppen eingenommen werden können. Endbenutzer-Gruppen können sein: Student, Developer, Tutor

Health Care Customer

AC-1-1-1	*Health Care Customer*
Beschreibung:	Health Care Customer sind diejenigen Benutzer, welche über eine Health Insurance Card verfügen. Sie stellen die einzusehenden Daten zur Verfügung, welche sich auf Ihrer Health Insurance Card befinden.

Health Care Provider

AC-1-1-2	*Health Care Provider*
Beschreibung:	Health Care Provider sind diejenigen Benutzer, welche über eine Health Professional Card verfügen. Sie stellen durch Ihre Health Professional Card die erforderlichen Rechte zur Verfügung, welche benötigt werden, um geschützte Inhalte auf der Health Insurance Card überhaupt einsehen zu können.

Developer

AC-1-2-1	*Developer*
Beschreibung:	Ein Student (Übungsteilnehmer) kann sowohl die Rolle eines Health Care Customers als auch die Rolle eine Health Care Providers übernehmen. Sein Ziel ist es, sich alle auf der Karte befindenden Informationen anzeigen zu lassen. Im speziellen interessieren ihn dabei die Möglichkeiten der Health Professional Card, sowie die Abläufe, die zum Anzeigen der Daten nötig sind.

Student

AC-1-2-2	*Student*
Beschreibung:	Ein Student (Übungsteilnehmer) kann sowohl die Rolle eines Health Care Customers als auch die Rolle eine Health Care Providers übernehmen. Sein Ziel ist es, sich die Informationen auf einer Health Insurance Card anzeigen zu lassen. Im speziellen interessieren ihn dabei die Möglichkeiten der Health Professional Card, sowie die Abläufe, die zum Anzeigen der Daten nötig sind. Die Health Insurance Card kann dabei seine eigene Karte oder eine für Demonstrationszwecke sein. Um auch medizinische Daten einsehen zu können, also die Rolle eines Health Care Providers einzunehmen, wird eine Health Professional Card benötigt. Diese kann beispielsweise durch den Tutor (Übungsleiter) zur Verfügung gestellt werden.

Tutor

AC-1-2-3	*Tutor*
Beschreibung:	Ein Tutor (Übungsleiter) kann ebenfalls sowohl die Rolle eines Health Care Customers als auch die Rolle eines Health Care Providers übernehmen. Sein Ziel ist es jedoch nicht, das System und seine Möglichkeiten kennenzulernen, sondern den Students (Übungsteilnehmern) das System vorzuführen. Ausserdem sollte er über eine Health Professional Card verfügen, um den Students (Übungsteilnehmern) den Zugriff auch auf medizinische Daten zu ermöglichen.

2.3 Personas

2.3.1 Developer Personas

PS-2-1	*Developer*
Aktor:	• AC-1-2-1: Developer
	Christian Moser (29) arbeitet bei einem Software-Hersteller für Praxisinformationssysteme. Er soll im Auftrag seines Arbeitgebers die Versichertenkarte in den Arbeitsablauf ihres Standardproduktes integrieren. Benutzung System: • In der Einarbeitungsphase • Fokus auf technische Implementierung „Ich möchte in möglichst kurzer Zeit möglichst viel über die technischen Details der Versichertenkarte erfahren.“

Table 1: Developer Personas

2.3.2 Student Personas

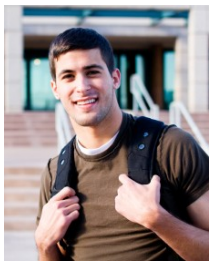
PS-2-2	*Student*
Aktor:	• AC-1-2-2: Student
	Marco Waser (24) ist Bachelor-Student in Informatik. Er möchte sein Wissen in Richtung Informationssysteme im Gesundheitswesen ausbauen. Benutzung System: • Wenige Male für Schulung • Usability weniger im Vordergrund • Ist interessiert an den technischen Abläufen „Informationssysteme im Gesundheitswesen interessieren mich, da dies ein zukunftsträchtiger Markt ist. Ich will mit Hilfe des Demonstrators möglichst viel über die elektronische Versichertenkarte lernen.“

Table 2: Student Personas

2.3.3 Tutor Personas

PS-2-3	*Developer*
Aktor:	• AC-1-2-3: Tutor
	*Prof. Dr. Günther Weiss* (57) ist Professor für medizinische Informationssysteme. Benutzung System: • Als Ergänzung zu seinen Vorlesungsunterlagen „Der Demonstrator soll mir helfen, meinen Studenten das System Versichertenkarte auf interaktive Weise näher zu bringen“

Table 3: Tutor Personas

Health Care Customer Personas

PS-1-1	*Health Care Customer*
Aktor:	• AC-1-1-1: Health Care Customer
	Frau Hochreutener (76) besucht regelmässig ihren Hausarzt. Mit neuer Technik hat sie wenig am Hut. „Früher ging es ja auch immer ohne Computer“, sagt sie. Sie hat früher als Telefonistin bei einer Versicherung gearbeitet. Benutzung System: • Regelmässig • Benutzung / Pin Eingabe soll möglichst einfach sein „Neue Technik, das ist nichts für mich. Für mich soll das System möglichst einfach zu bedienen sein, da ich regelmässig zum Arzt gehe.“

Table 4: Health Care Customer Personas

Health Care Provider Personas

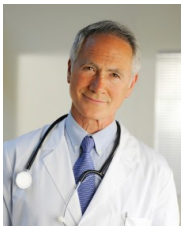
PS-1-2	*Health Care Provider*
Aktor:	• AC-1-1-2: Health Care Provider
	Dr. W. Isler (58) betreibt seit 20 Jahren seine eigene Arztpraxis. Er ist als Hausarzt tätig und hat sich ausserdem spezialisiert auf Tropenkrankheiten. Benutzung System: • Mehrmals pro Stunde • Hat sehr hohe Ansprüche an Bedienbarkeit des User Interface • Datenbearbeitung soll sehr effizient möglich sein • Das Fehlerrisiko soll möglichst gering gehalten werden „Ich bin einverstanden mit der Nutzung der Versichertenkarte zur Vereinfachung von administrativen Prozessen. Von der Speicherung medizinischer Notfalldaten halte ich allerdings nicht viel. Ich bin aber bereit, diese Daten auf Wunsch auf der Karte zu erfassen. Dies darf aber nicht mehr als 5 Minuten pro Patient in Anspruch nehmen.“

Table 5: Health Care Provider Personas

Der Demonstrator dient in erster Linie der Nutzung in der Lehre. Die beiden erfassten Personas *Dr. W. Isler* und *Frau Hochreutener* haben deshalb für diese Applikation eine geringere Bedeutung als die Personas *Marco Waser und Christian Moser.* Die an die reale Nutzung angelehnten Personas wurden jedoch in Betracht gezogen, damit die reale Nutzung der Applikation im Hinterkopf behalten werden kann. Dies soll zu einem möglichst realistischen Demonstrator führen.

2.4 Use Cases

2.4.1 Learn Possibilities of Health Insurance Card

UC-1	*Learn Possibilities of Health Insurance Card*
Aktoren:	• AC-1-2-1: Developer • AC-1-2-2: Student
Beschreibung:	Die sich auf der Health Insurance Card befindenden Daten anzeigen lassen.
Interessen:	Möglichkeiten der Health Insurance Card kennenlernen.
Vorbedingungen:	• Kartenlesegerät mit dem Computer verbunden • Health Insurance Card in Kartenleser eingeführt • Health Professional Card in Kartenleser eingeführt (optional)
Nachbedingungen:	• -
Hauptszenario:	1. Developer oder Student wählt einen bestimmten Datensatz aus 2. Datensatz wird von Health Insurance Card gelesen 3. Gelesener Datensatz wird angezeigt
Auftrittshäufigkeit:	Mindestens 21 mal während erster Verwendung des Demonstrators.

Table 6: Learn Possibilities of Health Insurance Card

2.4.2 Understand Processes Involved in Usage of Health Insurance Card

UC-2	*Understand Processes Involved in Usage of Health Insurance Card*
Aktoren:	• AC-1-2-1: Developer • AC-1-2-2: Student
Beschreibung:	Die sich auf der Health Insurance Card befindenden Daten anzeigen lassen.
Interessen:	Die Abläufe bei der Verwendung der Health Insurance Card und Health Professional Card verstehen.
Vorbedingungen:	• Kartenlesegerät mit dem Computer verbunden • Health Insurance Card in Kartenleser eingeführt • Health Professional Card in Kartenleser eingeführt (optional)
Nachbedingungen:	• -

Hauptszenario:	1. Developer oder Student wählt einen bestimmten, geschützten Datensatz aus
	2. Authentifizierung wird geprüft
	  1. Bei erfolgreicher Prüfung wird Datensatz von Health Insurance Card gelesen und angezeigt
	  2. Bei nicht erfolgreicher Prüfung wird Authentifizierung verlangt und erneut geprüft

Auftrittshäufigkeit:	Einmal während Zeitdauer der erstmaligen Verwendung des Demonstrators.

Table 7: Understand Processes Involved in Usage of Health Insurance Card

2.4.3 Display Data on Health Insurance Card

UC-3	*Display Data on Health Insurance Card*
Aktoren:	• AC-1-2-1: Developer • AC-1-2-2: Student • AC-1-2-3: Tutor
Beschreibung:	Die sich auf der Health Insurance Card befindenden Daten anzeigen lassen.
Interessen:	Informationen anzeigen lassen, die sich auf der Health Insurance Card befinden.
Vorbedingungen:	• Kartenlesegerät mit dem Computer verbunden • Health Insurance Card in Kartenleser eingeführt • Health Professional Card in Kartenleser eingeführt (optional)
Nachbedingungen:	• -
Hauptszenario:	1. Benutzer wählt einen bestimmten Datensatz aus 2. Datensatz wird von Health Insurance Card gelesen 3. Gelesener Datensatz wird angezeigt
Auftrittshäufigkeit:	Mehrmals während Zeitdauer der Verwendung des Demonstrators.

Table 8: Display Data on Health Insurance Card

2.4.4 Present Possibilities of Health Insurance Card

UC-4	*Present Possibilities of Health Insurance Card*
Aktoren:	• AC-1-2-3: Tutor
Beschreibung:	Die sich auf der Health Insurance Card befindenden Daten anzeigen lassen.
Interessen:	Den Students (Übungsteilnehmern) die Möglichkeiten der Health Insurance Card aufzeigen.
Vorbedingungen:	• Kartenlesegerät mit dem Computer verbunden • Health Insurance Card in Kartenleser eingeführt • Health Professional Card in Kartenleser eingeführt (optional)
Nachbedingungen:	• -
Hauptszenario:	1. Tutor wählt einen bestimmten Datensatz aus 2. Datensatz wird von Health Insurance Card gelesen 3. Gelesener Datensatz wird angezeigt
Auftrittshäufigkeit:	Mehrmals während Zeitdauer der Präsentation des Demonstrators.

Table 9: Present Possibilities of Health Insurance Card

2.4.5 Authenticate Health Care User

Dieser technische Use Case ist als Detaillierung einer fachlichen Anforderung und zur Erhöhung des allgemeinen Verständnisses zu verstehen. Die Authentifizierung ist zwar kein direktes Ziel des Benutzers aber eine wichtige und vom Benutzer explizit verlangte Kernfunktionalität des Systems. Da der Demonstrator auch Sicherheitsaspekte und -mechanismen aufzeigen soll.

UC-5	*Authenticate Health Care User*

Aktoren:	• AC-1: Health Care User
	  • AC-1-1: Health Care Customer
	  • AC-1-2: Health Care Provider

Beschreibung:	Der Health Care User (Customer oder Provider) Authentifiziert sich durch Eingabe seines PIN-Codes. Die PIN-Eingabe ist optional. Ohne Pin ist der User automatisch durch das Einführen der Karte in das Lesegerät authentifiziert.
Interessen:	Erfolgreiche Authentifizierung. Um auf geschützte Daten der Health Card zugreifen zu können, müssen sich sowohl der Health Care Customer als auch der Health Care Provider zuerst authentifizieren.
Vorbedingungen:	• Kartenlesegerät mit dem Computer verbunden • Health Card des Health Care Users wurde in den Kartenleser eingeführt
Nachbedingungen:	• Ist der PIN-Code korrekt, gilt der Health Care User als authentifiziert
Hauptszenario:	1. Health Care User gibt PIN-Code ein 2. Demonstrator fragt bei Health Card ob PIN korrekt 3. Health Card Prüft ob PIN korrekt
Auftrittshäufigkeit:	Vor jedem Zugriff auf geschützte Daten.

Table 10: Use Case Authenticate Health Care User

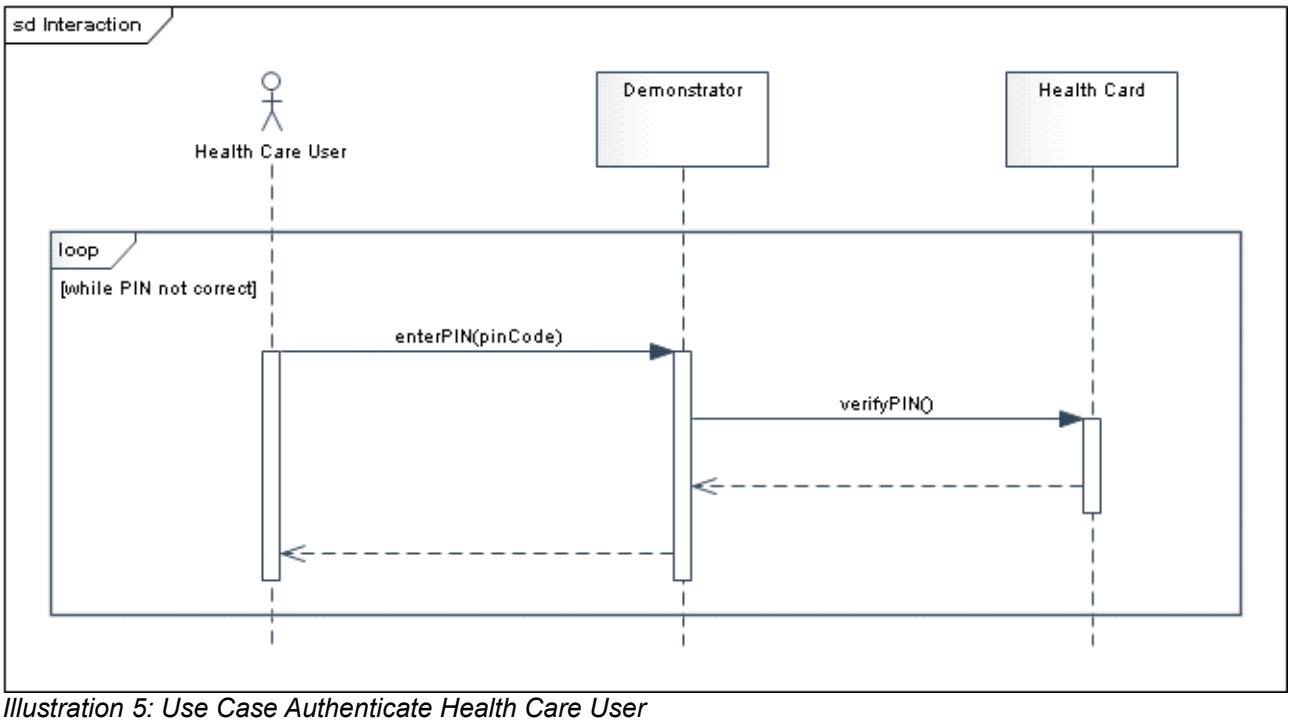


Illustration 5: Use Case Authenticate Health Care User

2.4.6 Display Health Insurance Card Identification Data

UC-6	*Display Health Insurance Card Identification Data*
Aktoren:	• AC-1-2: Health Care Provider
Beschreibung:	Der Health Care Provider liest die Kennnummer der Health Insurance Card.
Interessen:	Der Health Care Provider will die Kennnummer der Health Insurance Card herausfinden, da damit die Karte eindeutig identifiziert werden kann.
Vorbedingungen:	• Kartenlesegerät mit dem Computer verbunden • Health Insurance Card wurde in den Kartenleser eingeführt
Nachbedingungen:	• -
Hauptszenario:	1. Health Care Provider verlangt Identifikationsdaten der Health Insurance Card 2. Demonstrator liest Identifikationsdaten der Karte von Health Insurance Card
Auftrittshäufigkeit:	Mehrmals während Zeitdauer der Verwendung des Demonstrators. Wenn Health Care Provider Kennnummer der Health Insurance Card wissen will

Table 11: Use Case Display Health Insurance Card Identification Data

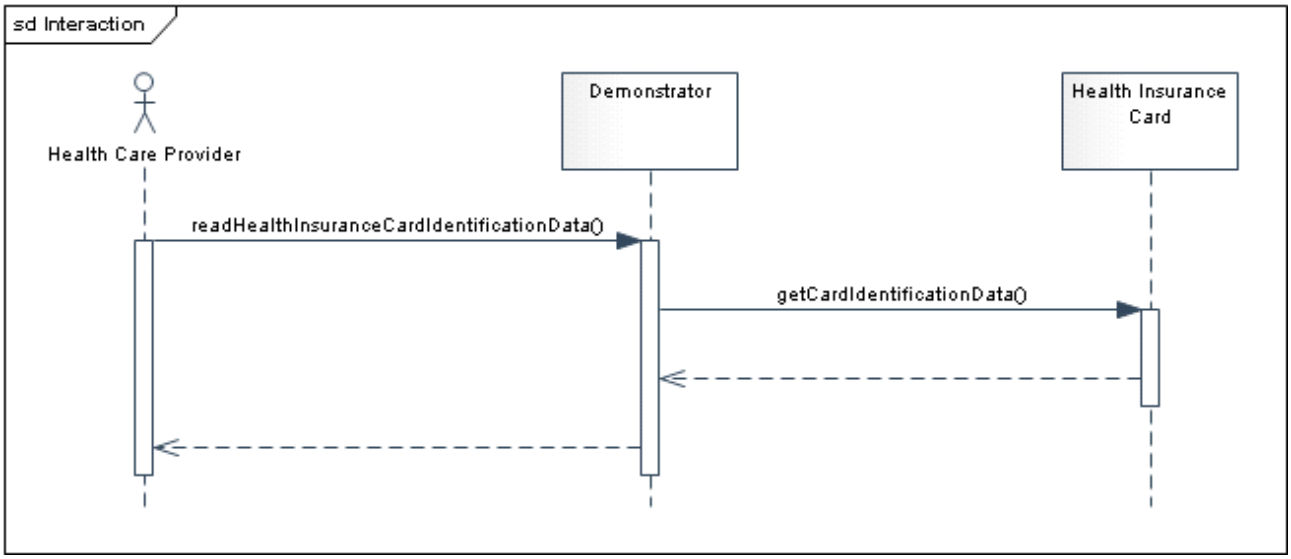


Illustration 6: Use Case Display Health Insurance Card Identification Data

2.4.7 Display Health Insurance Cardholder Identification Data

UC-7	*Display Health Insurance Cardholder Identification Data*
Aktoren:	• AC-1-2: Health Care Provider
Beschreibung:	Der Health Care Provider liest die Identifikationsdaten des Karteninhabers der Health Insurance Card.
Interessen:	Der Health Care Provider will die Identifikationsdaten des Karteninhabers (Health Care Customer) der Health Insurance Card herausfinden, da damit der Leistungsbezüger eindeutig identifiziert werden kann.
Vorbedingungen:	• Kartenlesegerät mit dem Computer verbunden • Health Insurance Card wurde in den Kartenleser eingeführt
Nachbedingungen:	• -
Hauptszenario:	1. Health Care Provider verlangt Identifikationsdaten des Health Insurance Cardholder 2. Demonstrator liest Identifikationsdaten des Karteninhabers von Health Insurance Card
Auftrittshäufigkeit:	Mehrmals während Zeitdauer der Verwendung des Demonstrators. Wenn Health Care Provider Identifikationsdaten des Karteninhabers der Health Insurance Card wissen will

Table 12: Use Case Display Health Insurance Cardholder Identification Data

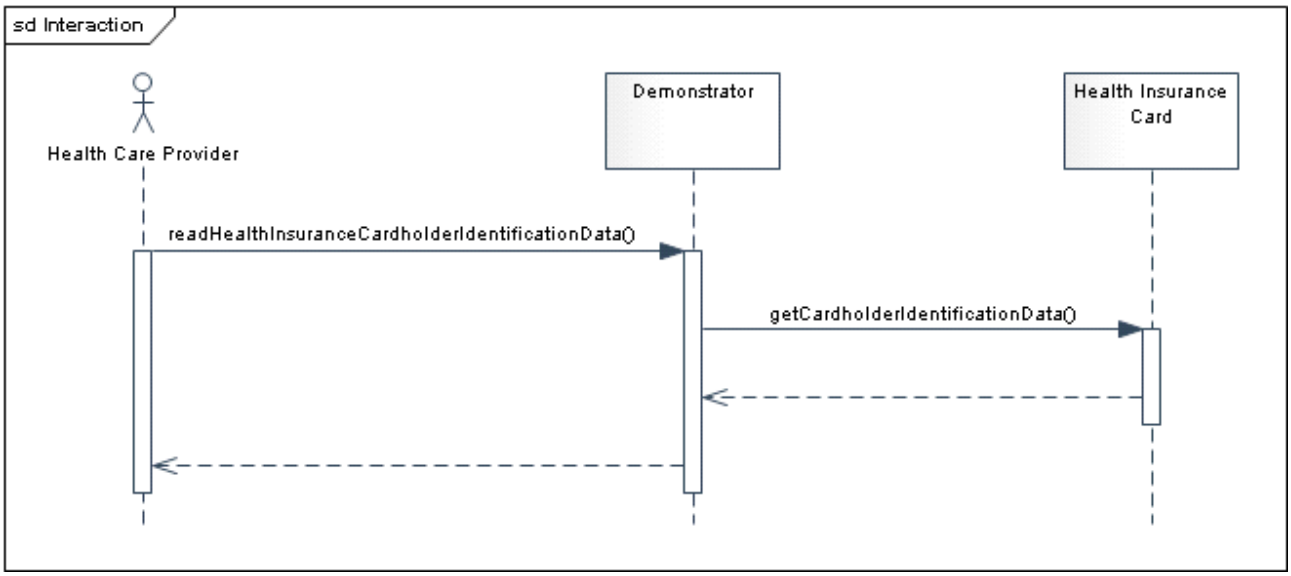


Illustration 7: Use Case Display Health Insurance Cardholder Identification Data

2.4.8 Display Health Insurance Cardholder Administrative Data

UC-8	*Display Health Insurance Cardholder Administrative Data*
Aktoren:	• AC-1-2: Health Care Provider
Beschreibung:	Der Health Care Provider liest die administrativen Daten des Karteninhabers der Health Insurance Card.
Interessen:	Der Health Care Provider will die administrativen Daten des Karteninhabers (Health Care Customer) der Health Insurance Card einsehen.
Vorbedingungen:	• Kartenlesegerät mit dem Computer verbunden • Health Insurance Card wurde in den Kartenleser eingeführt
Nachbedingungen:	• -
Hauptszenario:	1. Health Care Provider verlangt administrative Daten des Health Insurance Cardholder 2. Demonstrator liest administrative Daten des Karteninhabers von Health Insurance Card
Auftrittshäufigkeit:	Mehrmals während Zeitdauer der Verwendung des Demonstrators. Wenn Health Care Provider administrative Daten des Karteninhabers der Health Insurance Card wissen will

Table 13: Use Case Display Health Insurance Cardholder Administrative Data

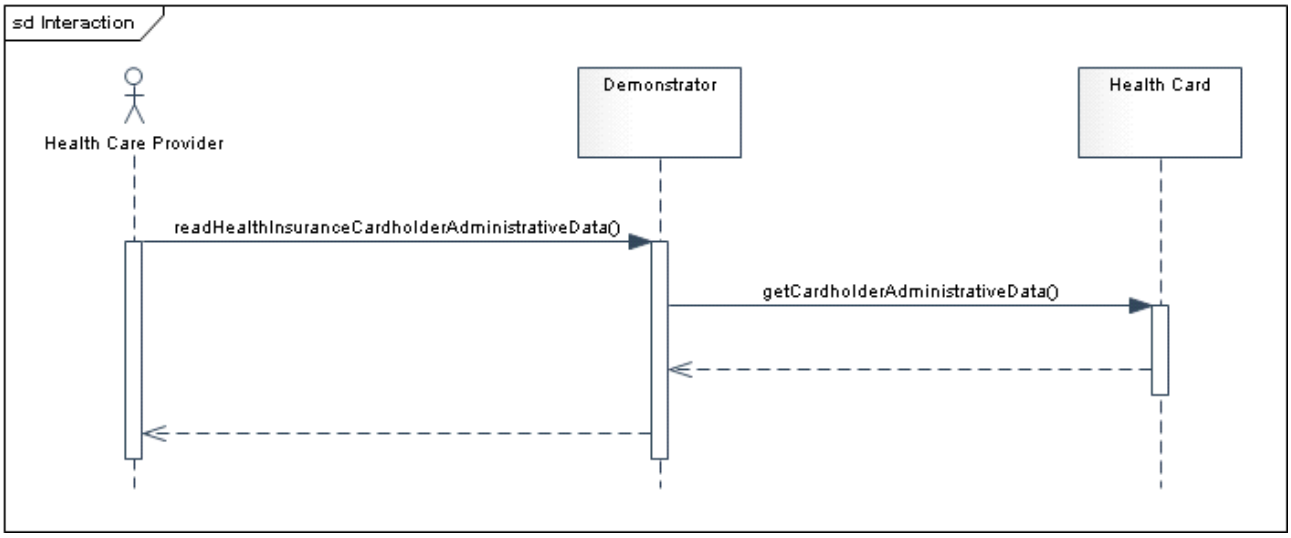


Illustration 8: Use Case Display Health Insurance Cardholder Administrative Data

Display Identification Data

UC-8-1	Display Identification Data
Beschreibung:	Der Health Care Provider liest die Identifikationsdaten des Health Care Customers von dessen Health Insurance Card.

Table 14: Use Case Display Identification Data

Display General Administrative Data

UC-8-2	Display General Administrative Data
Beschreibung:	Der Health Care Provider liest die allgemeinen administrativen Daten des Health Care Customers von dessen Health Insurance Card.

Table 15: Use Case Display General Administrative Data

Display Obligatory Health Insurance Coverage Data

UC-8-3	Display Obligatory Health Insurance Coverage Data
Beschreibung:	Der Health Care Provider liest die administrativen Daten zur obligatorischen Krankenpflegeversicherung des Health Care Customers von dessen Health Insurance Card.

Table 16: Use Case Display Obligatory Health Insurance Coverage Data

Display Obligatory Health Insurance Company Data

UC-8-4	Display Obligatory Health Insurance Company Data
Beschreibung:	Der Health Care Provider liest die administrativen Daten zum obligatorischen Krankenpflegeversicherer des Health Care Customers von dessen Health Insurance Card.

Table 17: Use Case Display Obligatory Health Insurance Company Data

Display Complementary Health Insurance Coverage Data

UC-8-5	Display Complementary Health Insurance Coverage Data
Beschreibung:	Der Health Care Provider liest die administrativen Daten zu den Zusatzversicherungen des Health Care Customers von dessen Health Insurance Card.

Table 18: Use Case Display Complementary Health Insurance Coverage Data

Display Complementary Health Insurance Company Data

UC-8-6	Display Complementary Health Insurance Company Data
Beschreibung:	Der Health Care Provider liest die administrative Daten zum Zusatzversicherer des Health Care Customers von dessen Health Insurance Card.

Table 19: Use Case Display Complementary Health Insurance Company Data

2.4.9 Display Health Insurance Cardholder Medical Data

UC-9	*Display Health Insurance Cardholder Medical Data*
Aktoren:	• AC-1-1: Health Care Customer • AC-1-2: Health Care Provider
Beschreibung:	Der Health Care Provider liest die medizinischen Daten des Karteninhabers der Health Insurance Card.
Interessen:	Der Health Care Provider will die medizinischen Daten des Karteninhabers (Health Care Customer) der Health Insurance Card einsehen.
Vorbedingungen:	• Kartenlesegerät mit dem Computer verbunden • Health Insurance Card wurde in den Kartenleser eingeführt • Health Care Customer erfolgreich authentifiziert • Health Care Provider erfolgreich authentifiziert
Nachbedingungen:	• -
Hauptszenario:	1. Health Care Provider verlangt medizinische Daten des Health Insurance Cardholder 2. Demonstrator liest medizinische Daten des Karteninhabers von Health Insurance Card
Auftrittshäufigkeit:	Mehrmals während Zeitdauer der Verwendung des Demonstrators. Wenn Health Care Provider medizinische Daten des Karteninhabers der Health Insurance Card wissen will

Table 20: Use Case Display Health Insurance Cardholder Medical Data

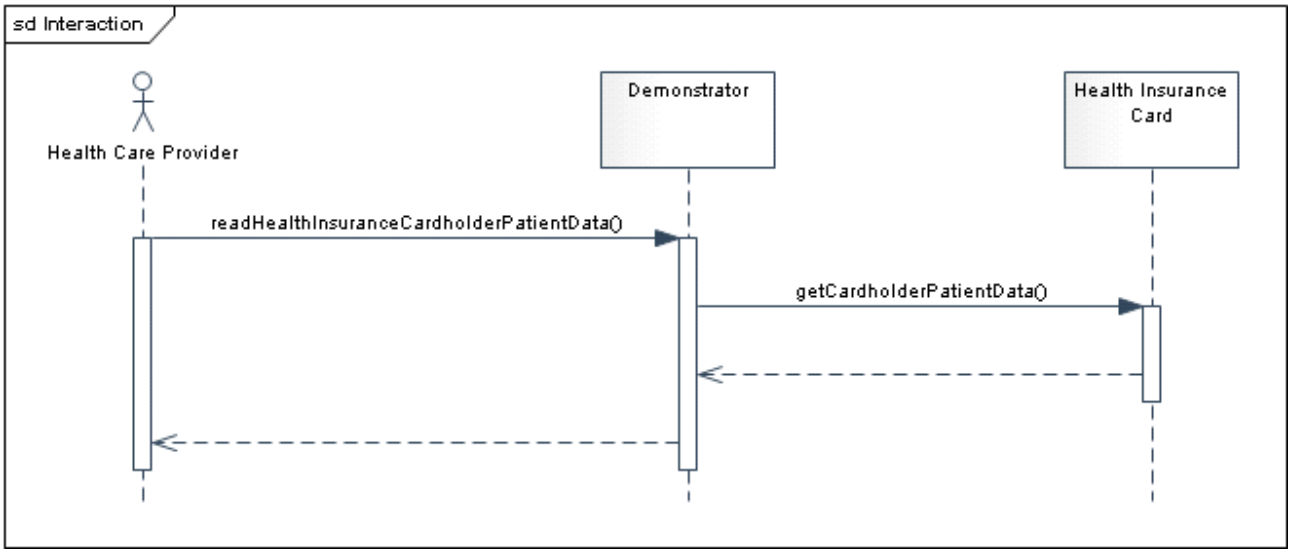


Illustration 9: Use Case Display Health Insurance Cardholder Medical Data

Display Disease Data

UC-9-1	*Display Disease Data*
Beschreibung:	Der Health Care Provider liest die Daten zu Krankheit und Unfallfolgen des Health Care Customers von dessen Health Insurance Card.

Table 21: Use Case Display Disease Data

Display Transplantation Data

UC-9-2	*Display Transplantation Data*
Beschreibung:	Der Health Care Provider liest die Daten zu Transplantationsdaten des Health Care Customers von dessen Health Insurance Card.

Table 22: Use Case Display Transplantation Data

Display Immediate Response Allergy Data

UC-9-3	*Display Immediate Response Allergy Data*
Beschreibung:	Der Health Care Provider liest die Daten zu Allergien mit Soforttypreaktionen des Health Care Customers von dessen Health Insurance Card.

Table 23: Use Case Display Immediate Response Allergy Data

Display Delayed Response Allergy Data

UC-9-4	*Display Delayed Response Allergy Data*
Beschreibung:	Der Health Care Provider liest die Daten zu 4. Allergien mit Spättypreaktionen des Health Care Customers von dessen Health Insurance Card.

Table 24: Use Case Display Delayed Response Allergy Data

Display Medication Data

UC-9-5	*Display Medication Data*
Beschreibung:	Der Health Care Provider liest die Daten zu Medikation des Health Care Customers von dessen Health Insurance Card.

Table 25: Use Case Display Medication Data

Display Vaccination Data

UC-9-6	*Display Vaccination Data*
Beschreibung:	Der Health Care Provider liest die Daten zu Impfungen des Health Care Customers von dessen Health Insurance Card.

Table 26: Use Case Display Vaccination Data

Display Blood and Transfusion Data

UC-9-7	*Display Blood and Transfusion Data*
Beschreibung:	Der Health Care Provider liest die Daten zu Blutgruppe und Transfusionen des Health Care Customers von dessen Health Insurance Card.

Table 27: Use Case Display Blood and Transfusion Data

Display Additional Data

UC-9-8	*Display Additional Data*
Beschreibung:	Der Health Care Provider liest die Daten zu zusätzlichen Einträgen des Health Care Customers von dessen Health Insurance Card.

Table 28: Use Case Display Additional Data

Display Contact Address Data

UC-9-9	*Display Contact Address Data*
Beschreibung:	Der Health Care Provider liest die Daten zu Kontaktadressen für den Notfall des Health Care Customers von dessen Health Insurance Card.

Table 29: Use Case Display Contact Address Data

Display Patient Decree Data

UC-9-10	*Display Patient Decree Data*
Beschreibung:	Der Health Care Provider liest die Daten zu Hinweise auf bestehende Patientenverfügungen oder Organspenderausweise des Health Care Customers von dessen Health Insurance Card.

Table 30: Use Case Display Patient Decree Data

3 Domain Modell

3.1 Domain Modell

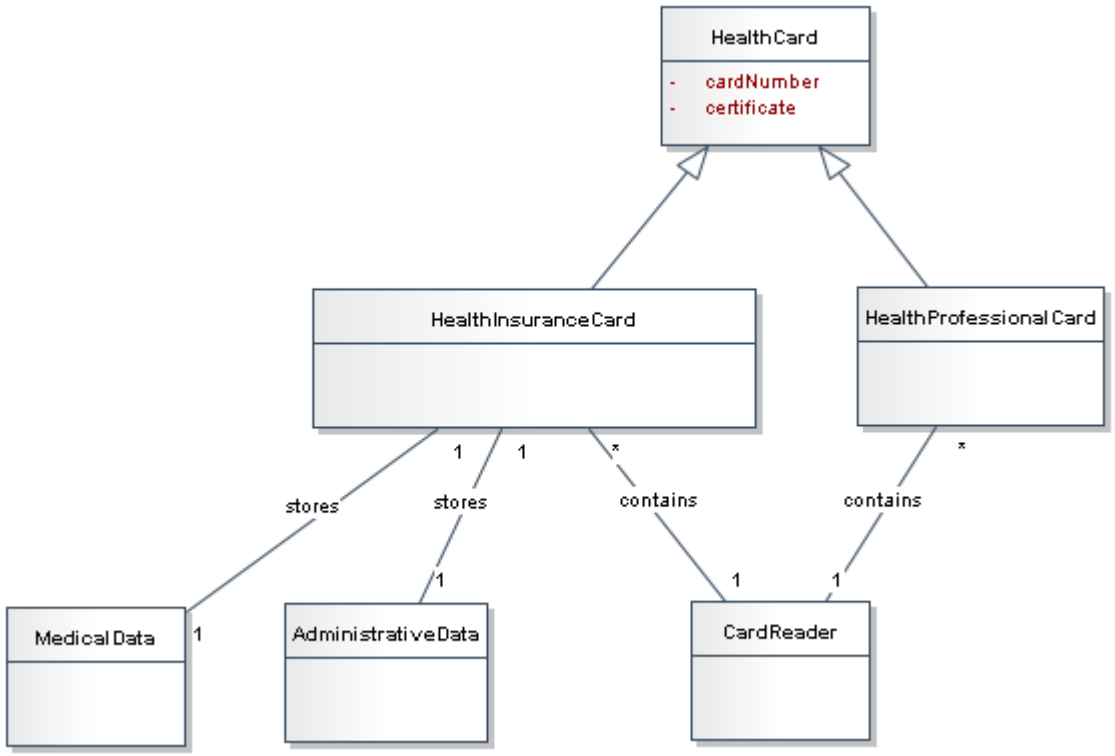


Illustration 10: Domain-Modell

Das Domain-Modell visualisiert die für den Demonstrator verwendeten Entitäten und deren Zusammenhänge. Hierbei liegt die Haupt-Komplexität nicht in den Zusammenhängen der Entitäten, sondern in den Datenobjekten. Diese Datenobjekte *MedicalData* und *AdministrativeData* werden auf den folgenden Seiten genauer beschrieben.

3.2 Datenmodell

Eine Kernfunktion des Demonstrators wird die Visualisierung der Datenstrukturen bzw. Daten-Definitionen sein. Aus diesem Grund wurden zum Domain Modell zusätzlich noch konzeptionelle Datenmodelle für administrative sowie medizinische Daten hinzugefügt.

3.2.1 Administrative Daten

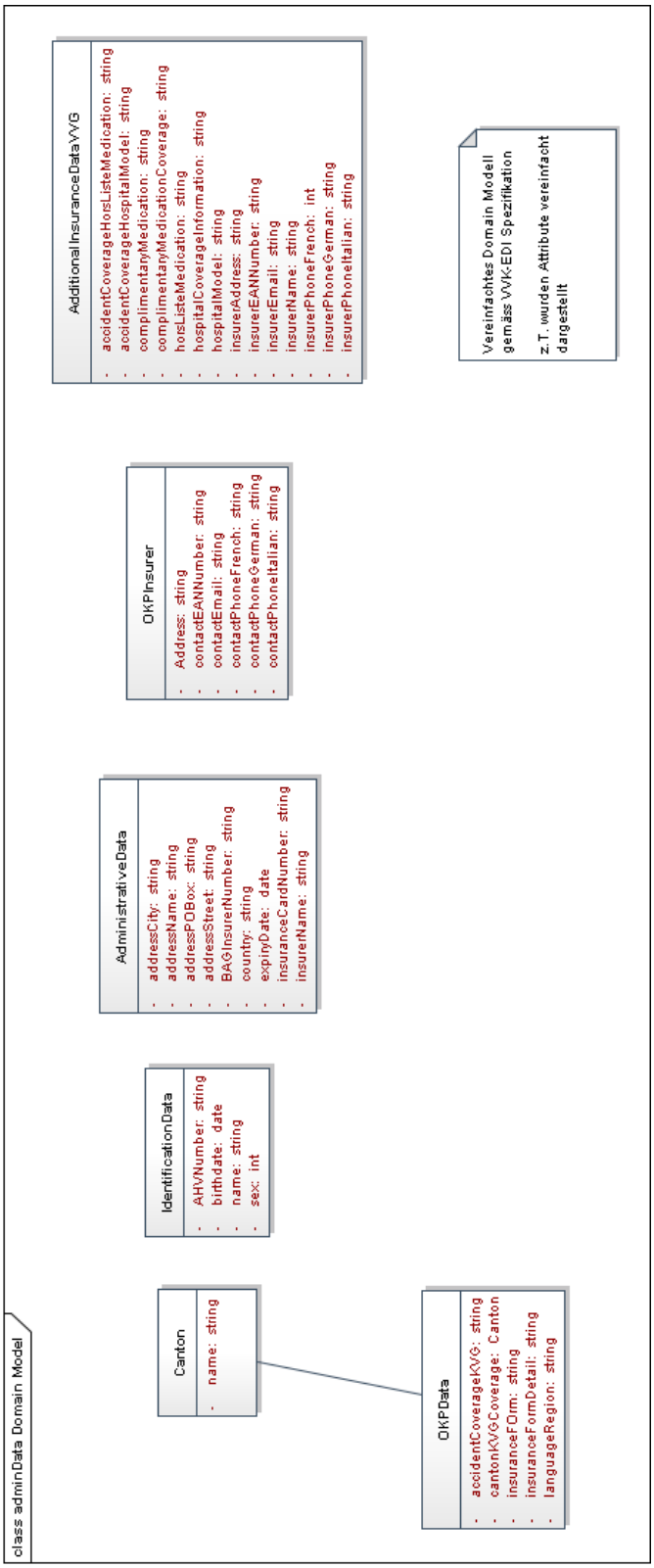


Illustration 11: Administrative Daten

3.2.2 Medizinische Daten

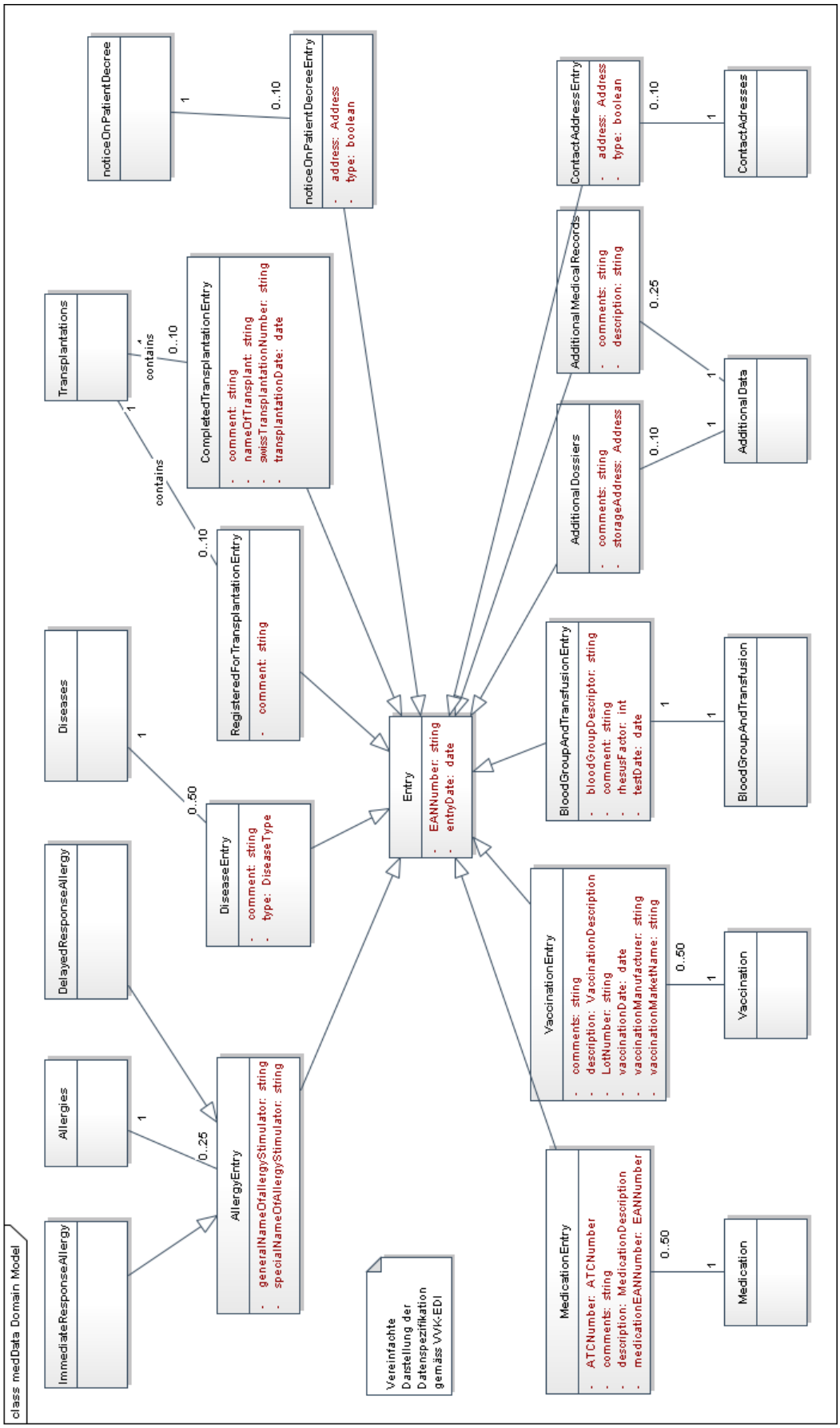


Illustration 12: Medizinische Daten

3.3 Konzeptbeschreibung / Objektkatalog

Das BAG hat Datenspezifikationen zu den administrativen und medizinischen Daten auf ihrer Website veröffentlicht.[1] Diese genau spezifizierten Daten liefern für die Demonstrator Applikation das Domain Modell. Die obigen Diagramme stellen diese Spezifikationen in leicht vereinfachter Form dar.
Es wird zwischen Administrativen Daten und Medizinischen Daten unterschieden.

3.3.1 Administrative Daten

Das Domain-Modell der administrativen Daten besteht aus einzelnen Datenklassen, die keine Referenzierungen unter einander haben.

Objekt	*Beschreibung*
IdentificationData	Beinhaltet Daten zum Karteninhaber
AdministrativeData	Beinhaltet administrative Daten zur Karte (Kartennr.) und zum Kartenherausgeber. Hinweis: Die Adressdaten sind gemäss VVK-EDI optional.
OKPInsurer	Angaben zum Versicherer (optional)
AdditionalInsuranceDataVVG	Daten zur Versicherungsabdeckung und Zusatzversicherungen. (optional)
Canton	Repräsentiert einen Abdeckungsbereich einer Versicherung (Schweizer Kanton oder Ausland)

Table 31: Domain-Objekte

1 http://www.bag.admin.ch/themen/krankenversicherung/07060/index.html?lang=de

3.3.2 Medizinische Daten

Die medizinischen Daten bestehen als Entries, die in einer bestimmten Anzahl zu einer verwaltenden Entität gehören (auf Chipkarte: EF mit Record Struktur und vorgegebener Anzahl Records). Im Domainmodell wurde dies berücksichtigt, indem die verwaltenden Entitäten (z.B. Medication, Vaccination) eine bestimmte Anzahl an Entries referenzieren. Entries leiten von einer Basisklasse ab, da zu jedem Entry bestimmte Attribute (EAN-Nummer der bearbeitenden Person, entryDate) gehören.

Objekt	*Beschreibung*
Entry	Basisklasse für Entries
Medication	Verwaltende Entität für Medikation (max. 50 Entries)
MedicationEntry	Beinhaltet Angaben zur einer bestimmten Medikation
Vaccination	Verwaltende Entität für Impfungen. (max. 50 Entries)
VaccinationEntry	Beinhaltet Angaben zum Impfstoff und Impfzeitpunkt
BloodGroupAndTransfusion	Verwaltende Entität für Blutgruppe und Transfusionen (1 Entry)
BloodGroupAndTransfusionEntry	Beinhaltet Angaben zu Blutgruppe, Datum des Tests, Anz. Transfusionen
AdditionalData	Verwaltende Entität für zusätzliche Einträge (in medizinisch begründeten Fällen) – Max 10 Entries zu medizinischen oder pharmazeutische Dossiers – Max 25 Entries zu weiteren medizinischen oder Pharmazeutischen Einträgen
AdditionalDossiers	Beinhaltet Angaben zu einem bestimmten medizinischen oder pharmazeutischen Dossier
AdditionalMedicalRecords	Beinhaltet Angaben zu weiteren medizinischen oder pharmazeutischen Einträgen
ContactAddresses	Verwaltende Entität für Kontaktadressen (für Notfall) Max. 10 Entries
ContactAddressEntry	Beinhaltet Angaben zu einer Kontaktadresse
Allergies	Verwaltende Entität für Allergien. – Immediate Response: Soforttypreaktionen (max. 25 Entries) – Delayed Response: Spättypreaktionen (max. 25 Entries)
AllergyEntry	Abstrakte Basisklasse für Allergie-Einträge
DelayedResponseAllergy	Beinhaltet Angaben zu einer Spättypreaktion
ImmediateResponseAllergy	Beinhaltet Angaben zu einer Soforttypreaktion
Diseases	Verwaltende Entität für Krankeiten und Unfallfolgen (max. 50 Entries)
DiseaseEntry	Beinhaltet die Bezeichnung der Krankheit/Unfallfolge und ein Bemerkungsfeld
Transplantations	Verwaltende Entität für Transplantationen – Zur Transplantation angemeldet (max. 10 Entries) – Bereits durchgeführte Transplantationen (max. 10 Entries)
RegisteredForTransplantationEntry	Beinhaltet ein Bemerkungsfeld zur angemeldeten Transplantation
CompletedTransplantationEntry	Beinhaltet Angaben zu einer bereits durchgeführten Transplantation.

Table 32: Domain-Objekte medizinische Daten

4 Anhang

4.1 Abbildungsverzeichnis

4.2 Tabellenverzeichnis

4.3 Literaturverzeichnis

Elektronische Versichertenkarte

Design

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-03-28
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Dokumentation des Designs des Demonstrators. In diesem Dokument sind auch Angaben zur konkreten Implementierung zu finden.

1.2 Zweck

Zweck ist die Dokumentation des Designs. Es sollen die Architektur, das externe Design, das Datenmodell, die verwendeten Technologien sowie weitere relevante Designentscheidungen festgelegt werden.

1.3 Gültigkeit

Dieses Dokument dient als Grundlage für den Demonstrator und hat deshalb Gültigkeit über die gesamte Dauer der Arbeit am Demonstrator. Änderungen werden laufend ergänzt und sind durch die Versionskontrolle ersichtlich.

2 Architektur

2.1 Schichtenarchitektur

Die Applikation wurde in 5 logische Schichten (Layers) unterteilt. Zugriffe finden grundsätzlich nur auf darunter liegende Schichten statt. Die Schichten können wiederum in 3 Gruppen unterteilt werden. In der Gruppe UI befinden sich sämtliche Schichten für das User Interface. In der Gruppe Logic befinden sich alle Schichten zur Steuerung des Applikationsflusses, Hilfsklasse und Funktionssammlungen. In der dritten Gruppe befinden sich die Schichten welche als API für die Arbeit mit der elektronischen Versichertenkarte verwendet werden können.

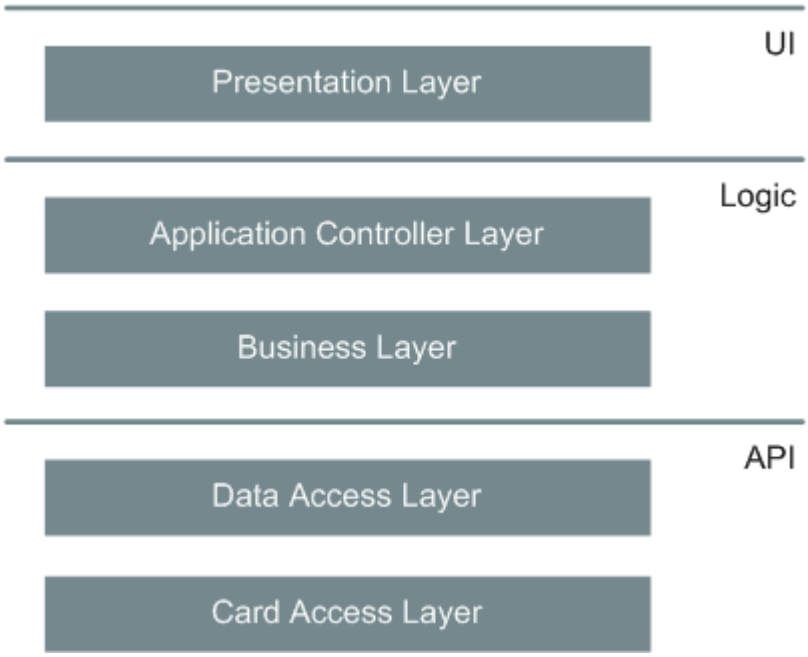


Illustration 1: Architekturübersicht

2.1.1 Presentation Layer

Entspricht der grafischen Benutzeroberfläche der Applikation. Sie erlaubt dem Benutzer die Interaktion mit dem Demonstrator über eine grafische Oberfläche.

2.1.2 Application Controller Layer

Controller für das User Interface. Steuert den Applikationsfluss. Kommuniziert mit den Datenobjekten des Business Layers. Überwacht auch Änderungen an der Hardware.

2.1.3 Business Layer

Separiert die Business Logic von den anderen Layern. Er beinhaltet die Domain-Klassen für den Demonstrator und greift über den Data Access Layer auf die Daten der Versichertenkarte zu. Weiter beinhaltet er Abstraktionen für Versichertenkarte, HPC und CardReader in Form von Wrapperklassen.

2.1.4 Data Access Layer

Stellt eine einfache Schnittstelle für den Low-Level Datenzugriff zur Verfügung. Die von der SmartCard gelesenen Rohdaten werden von Konvertern in spezifikationskonforme Datenobjekte umgewandelt. Die Datenobjekte entsprechen den XML-Spezifikationen des BAG und können in XML-Form serialisiert und deserialisiert werden.

2.1.5 Card Access Layer

Beinhaltet Klassen für den Low-Level Zugriff auf Daten und Funktionen der SmartCards. Hier werden Daten in Form von TLV-Collections oder Byte-Arrays zurückgegeben.

2.2 Physische Architektur

Der Demonstrator kann sowohl Standalone als auch in einem Webbrowser ausgeführt werden. Da die einzelnen Bestandteile des Demonstrators auch einzeln und direkt verwendet werden, wurde die Applikation in zwei Teile aufgeteilt, in UI/Logic und API. Das API kann dabei auch unabhängig vom User Interface verwendet werden.

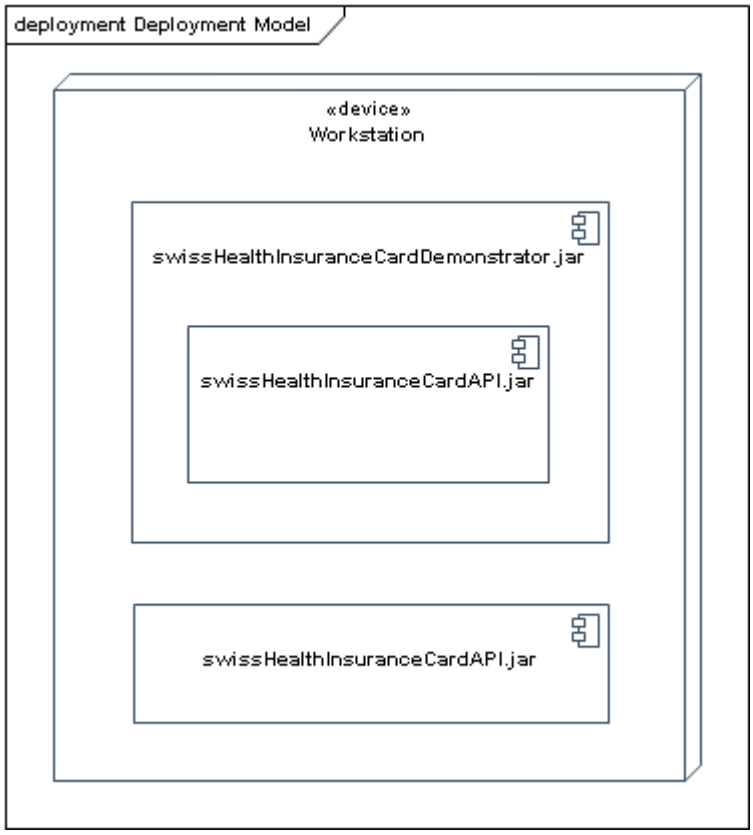


Illustration 2: Physische Architektur

2.3 Logische Architektur

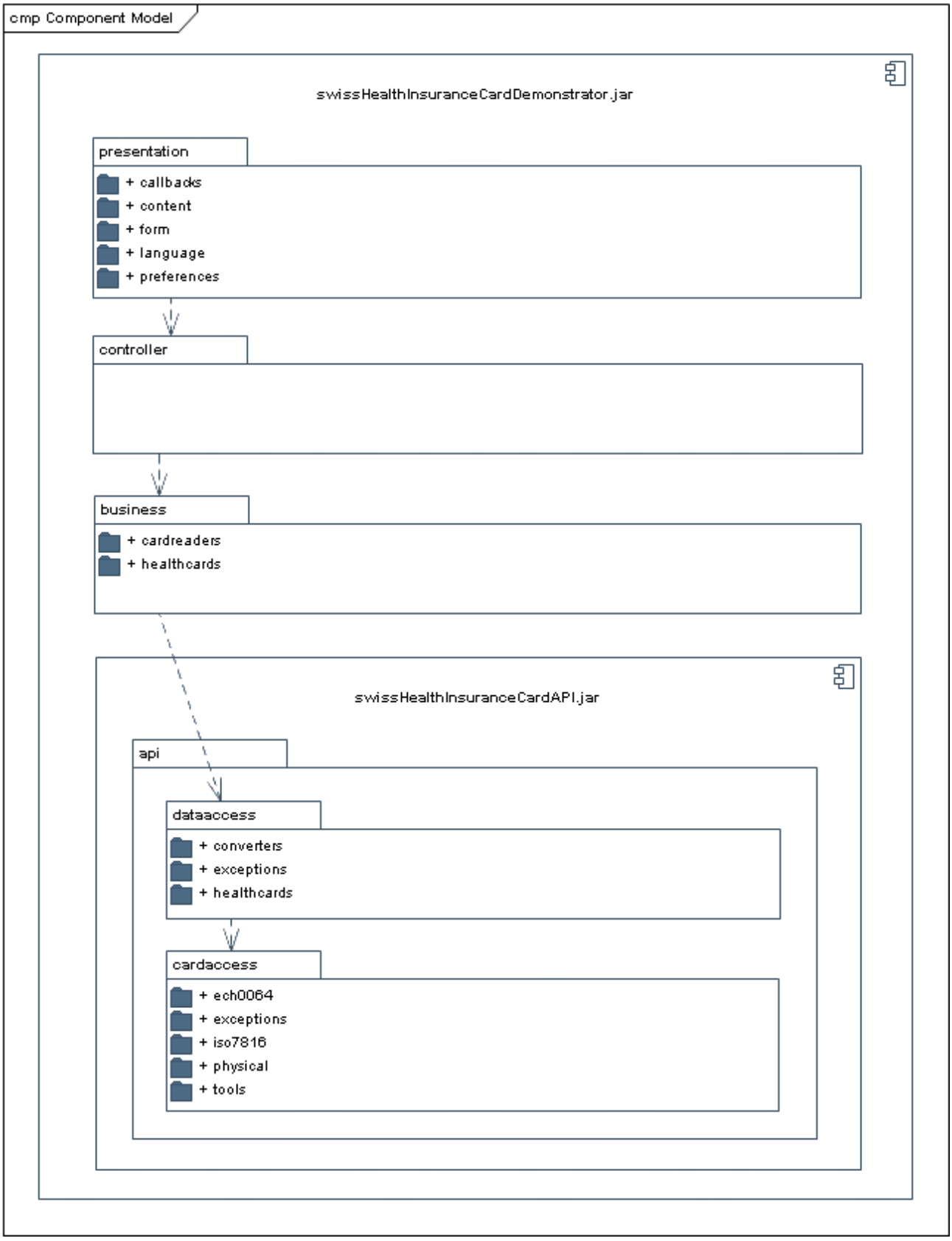


Illustration 3: Paketdiagramm

2.4 Pakete

Die nachfolgende Auflistung beschreibt die wichtigsten Pakete wie sie effektiv implementiert wurden. Wenn keine spezifischen Schnittstellen angegeben sind, bedeutet dies, dass keine zentrale Schnittstelle für das Paket vorhanden ist und direkt auf die entsprechenden Klassen zugegriffen werden kann.

2.4.1 Presentation

PK-1	*presentation*
Beschreibung:	Beinhaltet Klassen und weitere Pakete für die Benutzeroberfläche.
Struktur:	Jeder Bestandteil des User Interfaces ist in einer eigenen Klasse gekapselt.
Schnittstellen:	-

Table 1: Paket presentation

2.4.1.1 Content

PK-1-1	*content*
Beschreibung:	Beinhaltet Pakete für die Strukturierung und Generierung der im User Interface angezeigten Daten. Das Paket structure stellt eine Datenstruktur zur Verfügung, welche die im User Interface angezeigten Daten repräsentiert. Das User Interface wird aus dieser Datenstruktur generiert. Das Paket generator generiert eine solche Datenstruktur aus den Daten der Versichertenkarte.
Struktur:	Besteht aus den Subpaketen generator und structure.
Schnittstellen:	Das Paket content selber bietet keine Schnittstellen, da es lediglich die beiden Subpakete beinhaltet. Im Subpaket generator gibt es die Klasse *HealthInsuranceCardContentGenerator*, welche als Schnittstelle zum entsprechenden Subpaket fungiert. Im Subpaket structure gibt es die Klasse *HealthInsuranceCardContent*, welche als Schnittstelle zum entsprechenden Subpaket fungiert.

Table 2: Paket content

2.4.1.2 Preferences

PK-1-2	*preferences*
Beschreibung:	Im Paket Preferences befinden sich sämtliche Klassen zur Verwaltung der Programmeinstellungen.
Struktur:	-
Schnittstellen:	Über die Klasse *Configuration* können sämtliche Programmeinstellungen abgerufen werden. Programmeinstellungen umfassen Layouteinstellungen, Farbschemata, sowie Einstellungen zum Allgemeinen verhalten des User Interfaces.

Table 3: Paket preferences

2.4.1.3 Form

PK-1-3	*form*
Beschreibung:	Beinhaltet sämtliche Komponenten zur Erstellung von Formularen, wie Eingabefelder, Kontrollelemente etc.
Struktur:	Besitzt das Subpaket table zur Erstellung von Tabellen.
Schnittstellen:	-

Table 4: Paket form

2.4.1.4 Callbacks

PK-1-4	*callbacks*
Beschreibung:	Beinhaltet Interfaces und Klassen für Callback-Objekte, diese werden eingesetzt für Menü-Aufrufe und zum Verfolgen von Hardwareänderungen.
Struktur:	-
Schnittstellen:	-

Table 5: Paket callbacks

2.4.1.5 Language

PK-1-5	language
Beschreibung:	Alle Sprachbezogenen Klassen und Daten befinden sich in diesem Paket. Das Bündeln aller Spracheinstellungen erlaubt das einfache Ändern der Sprache des User Interfaces.
Struktur:	Im Subpaket properties sind sämtliche Sprach-Properties gespeichert. Für jede von der Applikation unterstützte Sprache und Region wird eine entsprechende Properties-Datei erstellt.
Schnittstellen:	Sämtliche Spracheinstellungen können über die Klasse *LanguageManager* abgerufen werden.

Table 6: Paket language

2.4.2 Controller

PK-2	controller
Beschreibung:	Beinhaltet Klassen für die Steuerung des Applikationsflusses und Verfolgung des Applikationszustandes.
Struktur:	Enthält keine Subpakete. Bietet Klassen zur Überwachung der Hardware auf Änderungen.
Schnittstellen:	Der *ApplicationController* speichert den aktuellen Zustand der Applikation und fungiert als ein Interface zu den darunterliegenden Schichten. Der ApplicationController sollte über die Klasse *ControllerFactory* erzeugt werden.

Table 7: Paket controller

2.4.3 Business

PK-3	business
Beschreibung:	Beinhaltet die Domain-Klassen und entsprechende Subpakete. Bietet Abstraktionen für den Hardwarezugriff.
Struktur:	Beinhaltet die zwei Pakete healthcards und cardreaders.
Schnittstellen:	Es existieren zwei Schnittstellen. Mit der HealthCardFactory können die verschiedenen Typen von HealthCards erzeugt werden. Mit *HealthCardPersistency* können HealthCards mittels XML persistiert oder geladen werden.

Table 8: Paket business

2.4.3.1 Healthcards

PK-3-1	healthcards
Beschreibung:	Enthält Abstraktionen der verschiedenen Health Cards in Form von Wrapperklassen, welche hardwarenähere Repräsentationen in tieferen Schichten der verschiedenen Health Cards umschliessen.
Struktur:	Es existieren Repräsentationen für Versichertenkarten (*HealthInsuranceCard* im Paket insurance) und HPC (*HealthProfessionalCard* im Paket professional).
Schnittstellen:	-

Table 9: Paket healthcards

2.4.3.2 Cardreaders

PK-3-2	cardreaders
Beschreibung:	Enthält eine Abstraktion eines CardReaders.
Struktur:	-
Schnittstellen:	-

Table 10: Paket cardreaders

2.4.4 API

PK-4	api
Beschreibung:	Beinhaltet die Pakete für den Zugriff auf Versichertenkarte, HPC und CardReader.
Struktur:	Besteht aus den zwei Paketen dataaccess und cardaccess.
Schnittstellen:	-

Table 11: Paket api

2.4.4.1 Data Access

PK-4-1	dataaccess
Beschreibung:	Herstellerunabhängige Abstraktion von low-level Kartenfunktionen. Bietet Klassen zur Konvertierung von Rohdaten in spezifikationskonforme Datenobjekte nach Spezifikation des BAG. Kann Datenobjekte in XML serialisieren und deserialisieren.
Struktur:	Beinhaltet die Subpakete healthcards, converters und exceptions.
Schnittstellen:	Bietet die Schnittstelle *CardFactory* für den Hardwarezugriff. Mit der *CardFactory* lassen sich die angeschlossenen Kartenlesegeräte sowie die physisch oder virtuell vorhandenen Versichertenkarten und HPCs in Form von Implementierungen der Schnittstellen *SwissHealthInsuranceCard* bzw. *SwissHealthProfessionalCard* zurückgeben. Die beiden Schnittstellen bieten abstrahierten Zugang zu Kartendaten- und Funktionen. Näheres siehe Javadoc. Diese Schnittstellen sowie die CardFactory bilden die Haupt-Schnittstelle zum API.

Table 12: Paket dataaccess

Healthcards

PK-4-1-1	healthcards
Beschreibung:	Beinhaltet Implementationen der SwissHealthProfessional- bzw. SwissHealthInsuranceCard-Interfaces.
Struktur:	Besteht aus den Subpaketen insurance und professional.
Schnittstellen:	-

Table 13: Paket healthcards

Converters

PK-4-1-2	converters
Beschreibung:	Bietet Klassen zur Serialisierung und Deserialisierung von Datenobjekten in XML sowie für die Umwandlung von Kartendaten zu Datenobjekten.
Struktur:	-
Schnittstellen:	Mit der Klasse *XMLConverter* können Datenobjekte in XML serialisiert/deserialisiert, persistiert und geladen werden. *TLVConverter* beinhaltet Methoden für das Umwandeln von TLV-codierten Kartendaten in Datenobjekte des Data Access Layers. Für eine Implementierung Schreiben / Lesen von medizinischen Daten müssten hier noch entsprechende Konverter-Methoden implementiert werden.

Table 14: Paket converters

Exceptions

PK-4-1-3	exceptions
Beschreibung:	Sammlung von Exceptions die im Paket dataaccess auftreten bzw. geworfen werden können.
Struktur:	-
Schnittstellen:	-

Table 15: Paket exceptions

2.4.4.2 Card Access

PK-4-2	cardaccess
Beschreibung:	Beinhaltet Klassen und Pakete für den Low-Level Zugriff auf Funktionen der SmartCards.
Struktur:	Beinhaltet die Subpakete physical, iso7816, ech0064, exceptions und tools.
Schnittstellen:	Mit der Klasse *LowLevelCardFactory* können die entsprechenden Implementierungen der zur Zeit verbundenen Health Cards geladen werden. Hier werden die angeschlossenen SmartCards automatisch enumeriert und die Klasse für die entsprechende Kartenimplementierung SASIS / Post instantiiert.

Table 16: Paket cardaccess

Physical

PK-4-2-1	physical
Beschreibung:	Klassen für den physischen Kartenzugriff. Sämtliche Schnittstellen zur Java SmartCard I/O API befinden sich in diesem Paket. Das sind einerseits CardCommunicator, andererseits CardReader .
Struktur:	Beinhaltet die Subpakete card, physical, und emulator.
Schnittstellen:	*CardCommunicator* ist eine Schnittstelle für die das Decorator Pattern[1] implementiert wurde. Es gibt eine konkrete Implementierung für das Senden von Kartenbefehlen sowie verschiedene Decorators z.B. für Debugging sowie einen einfachen Karten-Emulator für automatische UnitTests. *CardReader* dient als Adapter für die CardTerminal-Funktionen aus dem Java SmartCard I/O API. Diese Funktionen können mit dem dafür implementierten NullObject und dem Emulator ebenfalls für automatisierte UnitTests verwendet werden.

Table 17: Paket physical

ISO 7816

PK-4-2-2	iso7816
Beschreibung:	Beinhaltet Klassen zum Arbeiten mit ISO-7816-2-konformen SmartCards.
Struktur:	Beinhaltet das Subpaket data mit Datenobjekten und Collections für Low-Level Kartendaten.
Schnittstellen:	*BasicISO7816SmartCard* bietet vereinfachte Hilfsfunktionen und setzt APDU-Befehle an den referenzierten CardCommunicator ab. Diese Funktionen können genutzt werden, indem von BasicISO7816SmartCard abgeleitet wird (Siehe Paket ech0064).

Table 18: Paket iso7816

eCH-0064

PK-4-2-3	ech0064
Beschreibung:	Beinhaltet Klassen, die mit der Implementierung des eCH-0064 Standards in Verbindung stehen.
Struktur:	Beinhaltet die Subpakete data, hpc, pdc und misc.
Schnittstellen:	*SwissInsuranceCard* ist das Interface für den Zugriff auf low-Level Funktionen einer eCH-0064-konformen Versichertenkarte. Das Interface *HPC* dient als Schnittstelle zu Funktionen auf der Health Professional Card.

Table 19: Paket ech0064

1 Vgl. „Design Patterns: Elements of Reusable Object-Oriented Software“ Erich Gamma, Richard Helm, Ralph Johnson, John, Addison Wesley, 1994

Exceptions

PK-4-2-4	*exceptions*
Beschreibung:	Beinhaltet Exceptions, die auf diesem Layer geworfen werden können. Generell werden Exceptions bereits vom CardCommunicator abgefangen, so dass kaum mit Exceptions gearbeitet wird.
Struktur:	-
Schnittstellen:	-

Table 20: Paket exceptions

Tools

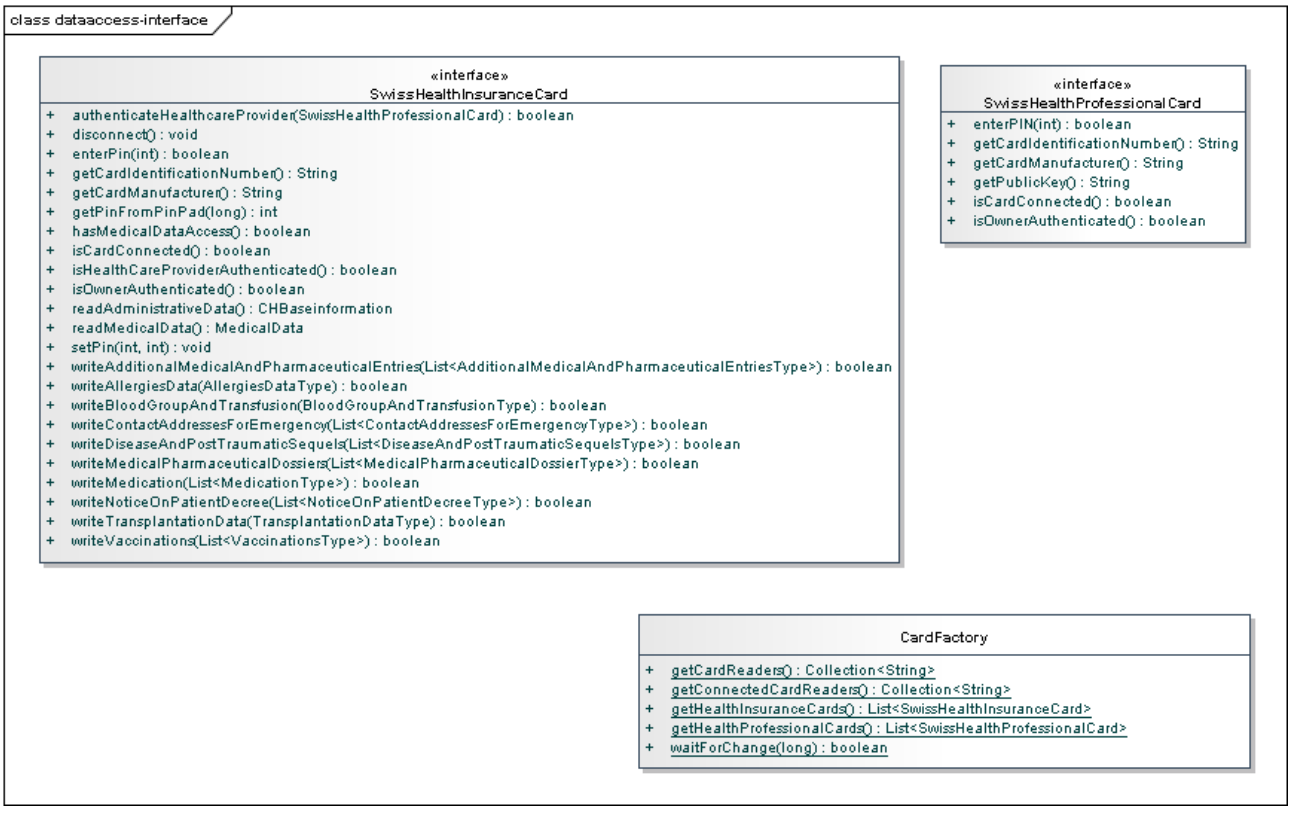
PK-4-2-5	*tools*
Beschreibung:	Beinhaltet Hilfsfunktionen zu Binäroperationen und Kryptologie-Operationen
Struktur:	-
Schnittstellen:	-

Table 21: Paket tools

2.5 Schnittstellenbeschreibungen

Für detaillierte Angaben zu den verschiedenen Schnittstellen siehe Javadoc. Die entscheidenden Schnittstellen der entsprechenden Pakete sind unter 2.4 Pakete zu finden. Im Folgenden eine kurze Übersicht zu den Schnittstellen des API.

2.5.1 Übersicht Schnittstellen des API



2.6 Architekturkonzepte

Die Softwarearchitektur wurde in Schichten aufgeteilt, damit die einzelnen Teile der Software einfach wiederverwendet oder weiterentwickelt werden können. Die Abhängigkeiten zwischen den einzelnen Schichten wurde möglichst gering gehalten. Da das API darauf ausgelegt ist, auch ohne User Interface verwendet zu werden, wurde die Kopplung zwischen User Interface und API auf ein Minimum reduziert.

Der Card Access Layer bietet Zugriffsmöglichkeiten auf die physisch angeschlossenen SmartCards. Dieser wurde so ausgelegt, dass über die gleiche Schnittstelle auch eine Implementation via PKCS#11 provider möglich wäre, was die parallele Operabilität mit anderen Kartendiensten (z.B. E-Mail Verschlüsselung im Mail-Client) erlauben würde.

Die Datenklassen im Business Layer entsprechen exakt den vom BAG vorgegebenen Spezifikationen. Diese Spezifikationen umfassen die Klassenstruktur, deren Namen und deren Aufbau. Die Spezifikationen sind definiert in den Dokumenten [DADV08] und [DMDV08]. Ausserdem existiert eine exakte XML-Spezifikation der administrativen und medizinischen Daten.

3 Internes Design

Im Folgenden werden einige wichtige Punkte des Internen Designs aufgeführt. Für eine detaillierte Ansicht des Internen Designs siehe Klassendiagramm in separater Datei.

3.1 SmartCardAccess

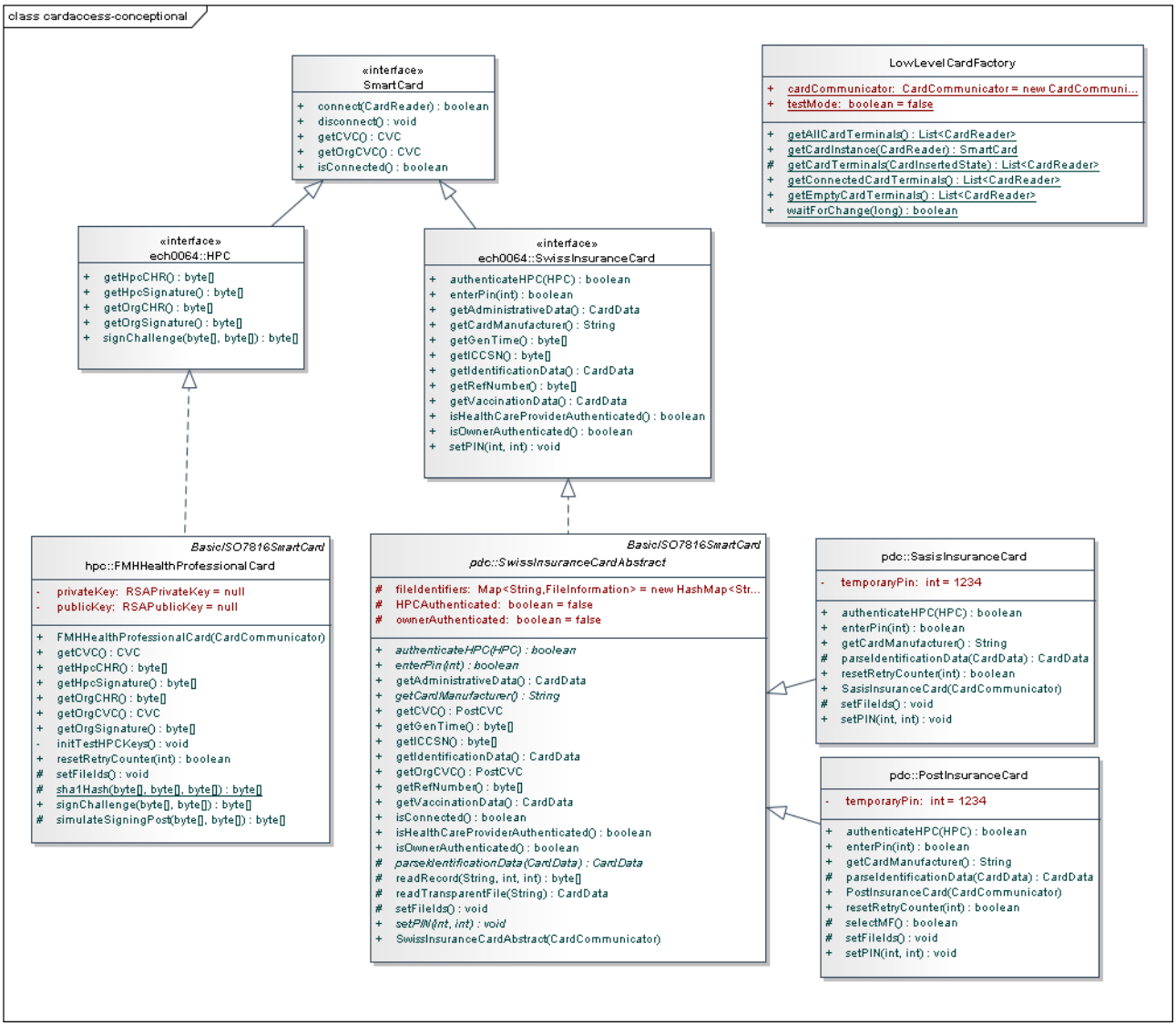


Illustration 4: Ansicht der eCH-0064 Implementierungsklassen

Der SmartCard Access Layer beinhaltet einerseits die Implementierung des eCH-0064-Standards, andererseits wird hier auch der physische Zugriff auf ISO-7816-konforme SmartCards durchgeführt. Über die LowLevelCardFactory werden angeschlossene Kartenlesegeräte, Versichertenkarten und HPCs erkannt und zurückgegeben.

Operationen nach eCH-0064 wurden entweder in der SwissInsuranceCardAbstract-Klasse implementiert, oder in einer davon abgeleiteten, herstellerspezifischen Klasse.

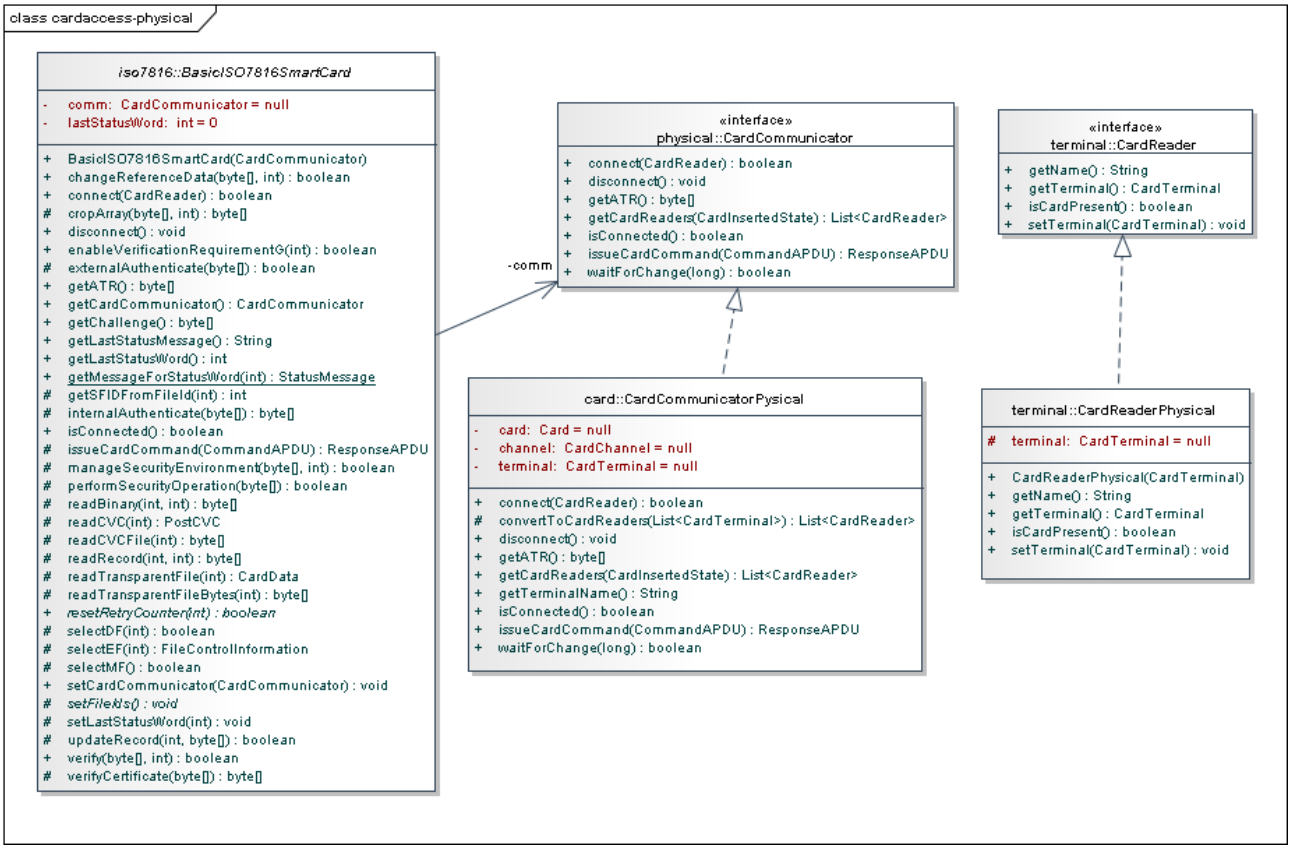


Illustration 5: Klassen für physischen Kartenzugriff

Sämtliche Kommunikation mit den SmartCards läuft über die Klasse *CardCommunicatorPhysical*. Für das *CardCommunicator* Interface wurden verschiedene Decorators implementiert, die für Debugging und für die Emulation von Kartenbefehlen genutzt werden können. Funktionen, welche die am System angeschlossenen Kartenlesegeräte betreffen, wurden ebenfalls im *CardCommunicator* implementiert. Für ein Kartenlesegerät wurde die Wrapper-Klasse CardReader eingeführt, damit die Abhängigkeit der Java SmartCard I/O innerhalb des cardaccess Paketes minimiert wird und damit auch Kartenlesegeräte emuliert werden können.

Die abstrakte Klasse *BasicISO7816SmartCard* beinhaltet Vereinfachungen von Kartenbefehlen. So kann z.B. mit selectDf(dfid) ein Verzeichnis auf der SmartCard selektiert werden, ohne Kenntnisse des dazu nötigen APDU-Kommandos zu haben.

4 Systemsequenzdiagramme

4.1 Authentifizierung Leistungserbringer

Die Authentifizierung von Leistungserbringern ist relativ umfangreich. Ausserdem existieren Inkompatibilitäten zwischen den verschiedenen Kartenimplementierungen. Der Vorgang der Authentifizierung eines Leistungserbringers ist im folgenden Diagramm dargestellt.

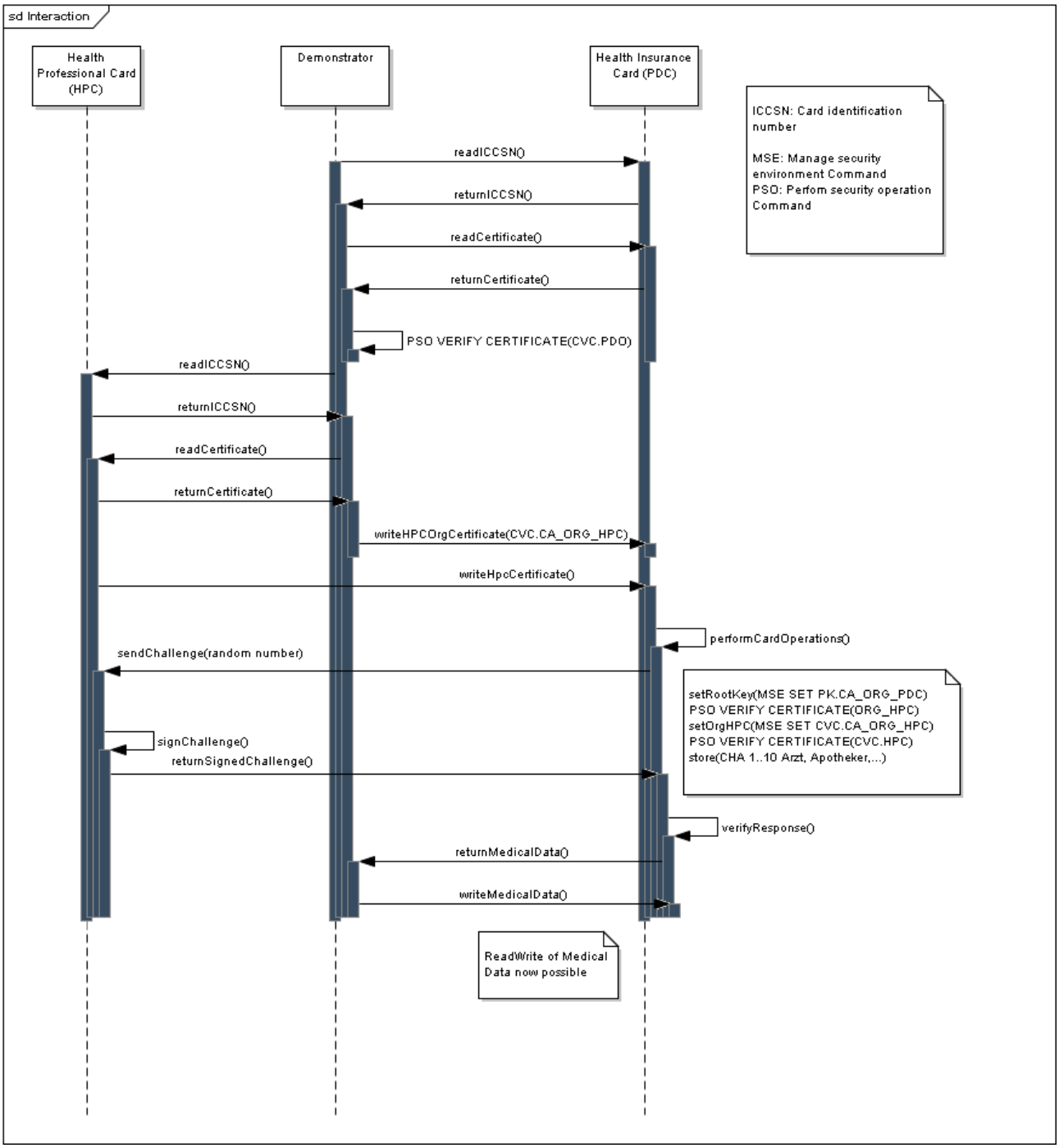


Illustration 6: Systemsequenzdiagramm Authentifizierung Leistungserbringer

Bei der ersten Benutzung einer Versichertenkarte muss neben dem HPC-Zertifikat auch das HPC-Root-Zertifikat in die Versichertenkarte importiert werden. Gemäss Post/SwissSign Detailspezifikation ist es auch möglich, das Challenge-Response-Verfahren auf beide Seiten durchzuführen, d.h. dass sich die Versichertenkarte auch bei der HPC authentifiziert. Dieser Schritt wurde jedoch nicht in der Spezifikation eCH-0064 festgelegt und ist daher optional.

4.2 Abruf administrative Daten vom UI aus durch alle Layers

Das folgende Systemsequenzdiagramm verdeutlicht den Ablauf eines Aufrufes durch alle Layer hindurch, am Beispiel von administrativen Daten.

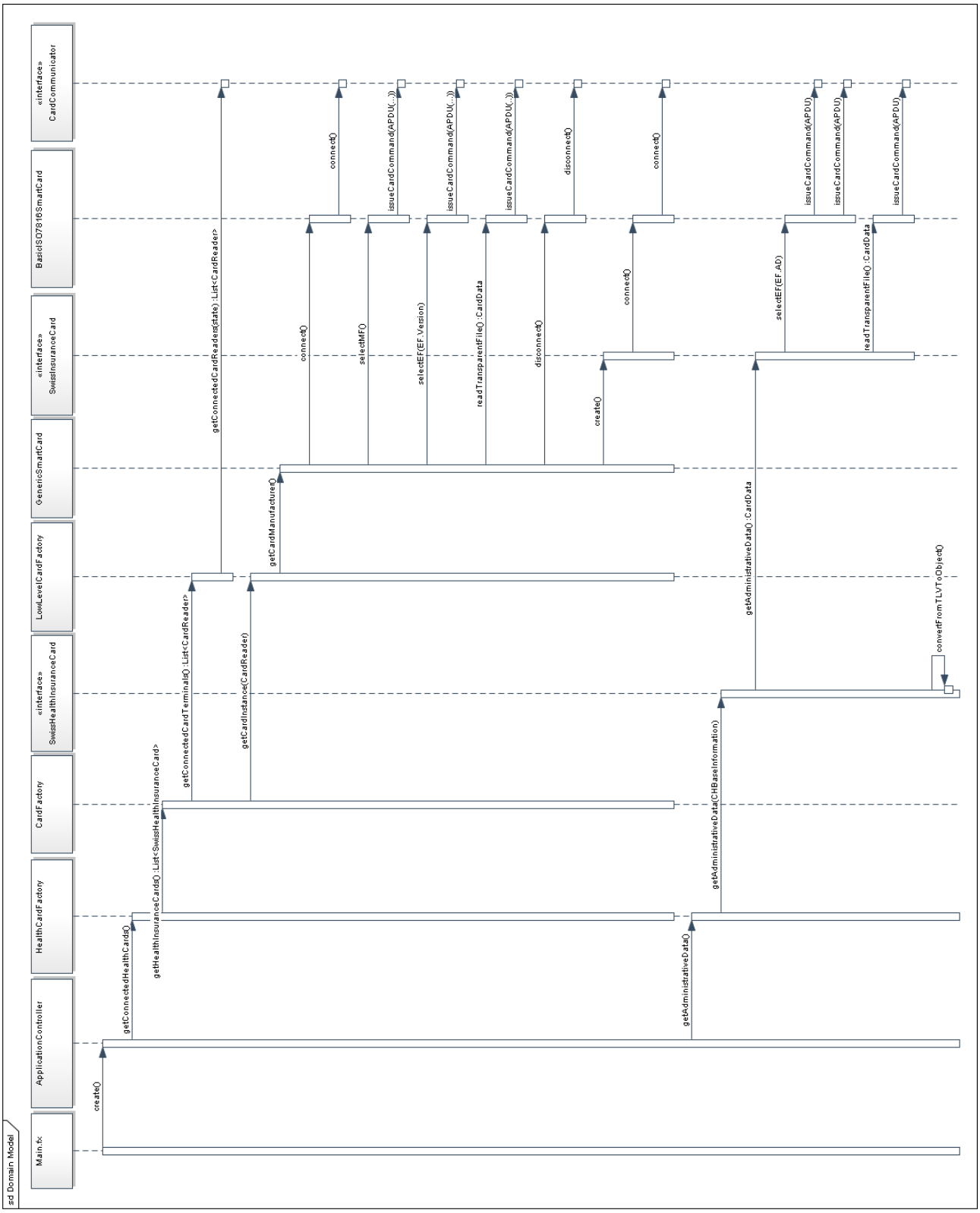


Abbildung 7: Systemsequenzdiagramm Aufruf durch alle Layer hindurch

Main.fx instantiiert den *ApplicationController*. Dieser wiederum ruft die *HealthCardFactory* des Business Layers auf, der über das darunter liegende API die angeschlossenen Karten aufruft. Innerhalb des API wird über ein *GenericSmartCard*-Objekt die Kartenimplementierung erkannt und die richtige Low-Level Klasse instantiiert. Die gesamte Hardware-Kommunikation läuft über die *CardCommunicator*-Klasse.

4.3 Gestaltung Benutzerschnittstelle

Im Folgenden der Grundlegende schematische Aufbau der Benutzerschnittstelle mit den verschiedenen Elementen.

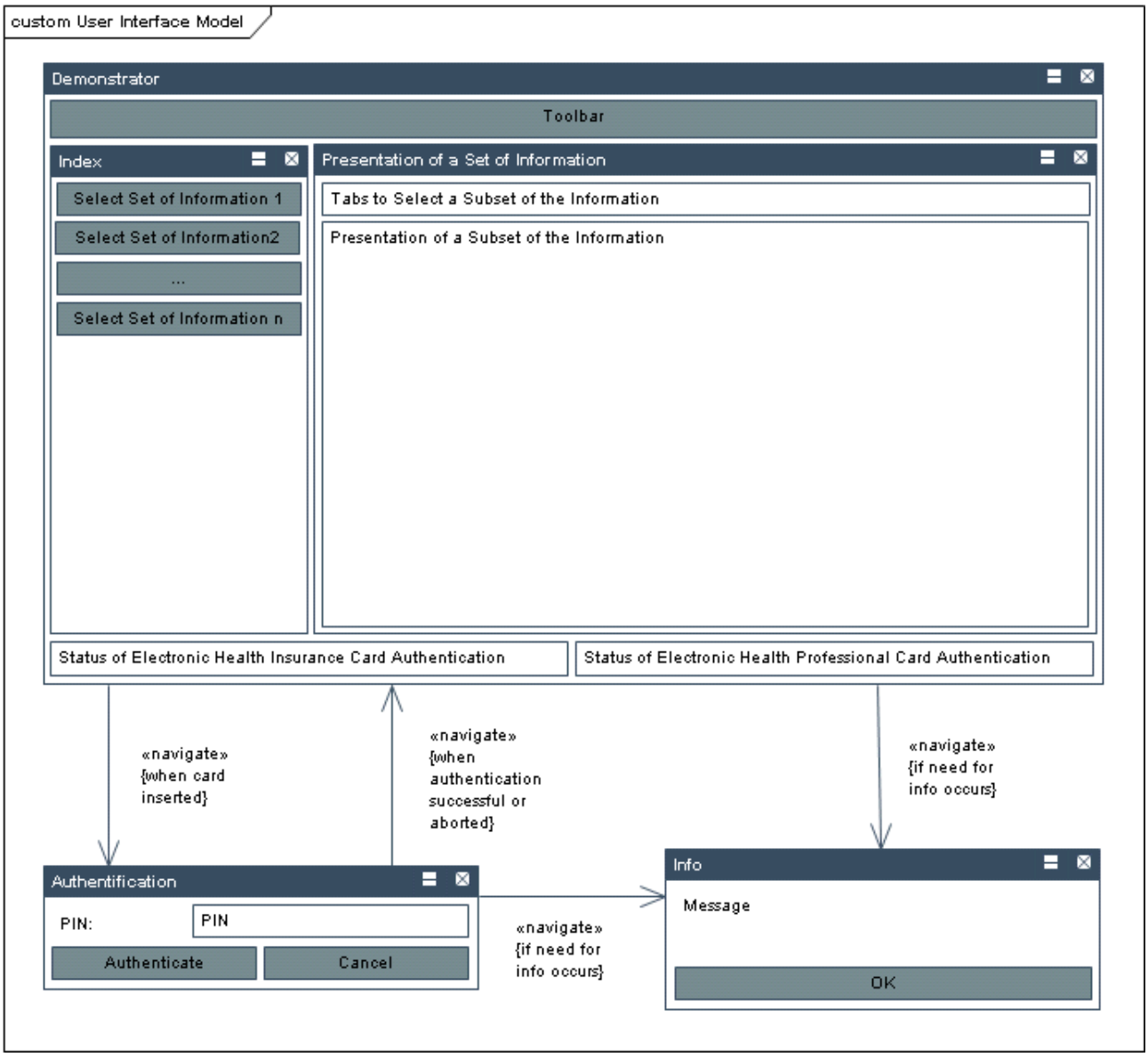


Illustration 8: User Interface Model

4.4 Implementierung Benutzerschnittstelle

Die Benutzerschnittstelle soll Standalone und eingebunden in eine Website verwendbar sein. Um beide Kriterien optimal zu erfüllen, ist die Benutzerschnittstelle in JavaFX implementiert. Genaueres zur Evaluation der Benutzerschnittstellen-Technologie wurde im Dokument „Technischer Bericht“ im Abschnitt „Umsetzungskonzept“ beschrieben.

5 Anhang

5.1 Abbildungsverzeichnis

5.2 Tabellenverzeichnis

5.3 Literaturverzeichnis

DADV08: BAG, Darstellung persönliche (medizinische) Daten Versichertenkarte, 2008
DMDV08: BAG, Darstellung persönliche (medizinische) Daten Versichertenkarte, 2008

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *Design Patterns. Elements of Reusable Object-Oriented Software.* Addison Wesley, 1994, ISBN 0-201-63361-2

Elektronische Versichertenkarte

Implementierung

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-06-10
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Dieses Dokument enthält Informationen zur konkreten Implementierung des Demonstrators, zu den eingesetzten Test-Methoden und Informationen für eine mögliche Weiterentwicklung des Programms.

1.2 Zweck

Verschaffen eines Überblicks über die konkrete Implementierung, die eingesetzten Testverfahren und Weiterentwicklungspunkte der Applikation.

1.3 Gültigkeit

Gültig während der gesamten Projektdauer.

2 Implementierung

2.1 Projektstruktur

Das Projekt SwissInsuranceCardDemonstrator wurde in zwei Subprojekte unterteilt:

- SwissInsuranceCardDemonstrator
  - SwissInsuranceCardAPI
  - SwissInsuranceCardUI

Hier ist zu bemerken, dass das UI-Projekt nicht nur den User-Interface-Code enthält, sondern auch die Logik des Benutzerinterfaces. Das API-Projekt beherbergt, wie bereits im Designdokument beschrieben, eine eigenständige Sammlung von Klassen für den Zugriff auf Daten und Funktionen der Schweizerischen Versichertenkarten bzw. HPCs.

2.2 Erläuterungen konkreter Klassen

Im Design-Dokument wurden bereits die konkrete Implementierung der einzelnen Pakete sowie der wichtigsten Klassen beschrieben. Für Details kann die Javadoc von API und User Interface konsultiert werden.

2.3 Build-Prozess

Die Builds wurden mit Maven automatisiert auf einem dafür eingerichteten Linux-Server durchgeführt. Die Pom.xml-Dateien sind auf der Abgabe-CD enthalten.

Folgende Maven-Plugins wurden verwendet:

Name	*Beschreibung*
maven-javadoc-plugin	Generierung der Javadoc.
maven-jarsigner-plugin	Signierung von JAR-Dateien. Diese wurden mit einem selbst erzeugten Key signiert, der 6 Monate gültig ist. Die JAR-Dateien mussten signiert sein, da sonst über WebStart kein Zugriff auf Dateien und Hardware möglich ist.
javafxc-maven-plugin	Plugin von easytesting.org für die Kompilierung von JavaFX-Sources zusammen mit Java-Sources. Dazu werden *.fx Sources an den auf dem Server installierten javafx-Compiler geschickt d.h. auf dem Server muss die JavaFX Runtime installiert sein. Das Tool befindet sich noch in einem frühen Entwicklungsstadium.

Table 1: Verwendete Maven-Plugins

Eine besondere Herausforderung war das Kompilieren von JavaFX-Source zusammen mit herkömmlichem Java-Quellcode. Grundproblem ist, dass JavaFX Sources vom JavaFX-Compiler in Java-Code übersetzt werden müssen. Dieser Compiler darf aus Lizenzgründen nur von Sun/Oracle angeboten werden, was eine Integration in Maven erschwert, da der Compiler nicht in Maven-Repositories angeboten werden darf. Beim Buildvorgang muss ein ANT-Task ausgeführt werden, welcher die .fx Dateien an javafxc, den Compiler für JavaFX sendet. In Eclipse und NetBeans existieren Plugins für diesen Vorgang. Da JavaFX noch relativ neu ist, gibt es dafür in Maven noch keine ausgereifte Lösung. Es wurde ein Plugin von Jfrog getestet, jedoch hat dies nicht richtig funktioniert und .fx Sources hätten in einem gesonderten Projekt abgelegt werden müssen. Das javafxc-maven-plugin verfolgt hier einen einfacheren Ansatz, der auch besser funktioniert hat. Jedoch waren häufig die erzeugten JAR-Files am Ende nicht lauffähig, da keine Einstiegsmethode gefunden wurde. Das Problem konnte nicht richtig reproduziert werden und wurde in Absprache mit den Entwicklern des Plugins als Bug gemeldet.

So musste trotz Server-Builds häufig lokal eine WebStart-fähige Version kompiliert und auf den Server kopiert werden. Build/Test des API-Projektes hat hingegen immer einwandfrei funktioniert. Aus diesem Grund wurden die Server-Builds vorallem für die automatische Javadoc-Erzeugung, Test sowie für das Metrik-Tool Sonar eingesetzt.

2.4 Verwendete Tools

Name	*Beschreibung*
Eclipse	Eclipse wurde für die Entwicklung eines Grossteils der Applikation verwendet.
NetBeans	Für die Entwicklung des JavaFX-basierten UserInterfaces wurde teilweise NetBeans benutzt, da dessen JavaFX-Unterstützung weiter ausgereift ist, als bei Eclipse.
JUnit	Für automatisierte Tests der API-Klassen wurde JUnit eingesetzt.
Maven	Für Server-basierte Builds wurde Maven mit verschiedenen Plugins eingesetzt.
JSmartCard Explorer	Tool für das Abschicken von SmartCard APDU-Befehlen. Wurde sehr häufig eingesetzt, um die spezifizierten Befehle zu überprüfen. http://www.primianotucci.com/
Jaxb	Teile der Datenklassen wurden mit dem Generator von Jaxb aus den XML-Schemas des Bundes generiert.
Subversion	Der Quellcode sowie die Dokumente des Projektes wurden auf einem auf dem Entwicklungsserver eingerichteten Subversion-Server gehostet.
Sonar	Code-Statistiken und Metriken. Wurde in den Maven-Buildprozess integriert.
Trac	Für issue-Tracking und Übersicht über den Entwicklungsstand wurde Trac eingesetzt.
MediaWIKI	Wiki für Sitzungsprotokolle, Wochenberichte und allgemeine Notizen.
Apache/mySQL	Apache mit diversen Zusatzmodulen. Hosting von SVN über http, Sonar über eine Tomcat-Integration, MediaWiki und Trac.

Table 2: Auflistung der verwendeten Tools im Entwicklungsprozess

3 Tests

3.1 Automatisiert

Für automatisierte Tests wurde JUnit 4 eingesetzt. Besonders die Klassen im API wurden durch UnitTest geprüft. Die Funktionen der Benutzeroberfläche wurden nicht automatisiert getestet. Das lag einerseits daran, dass automatisiertes Testen von Benutzerinteraktionen generell schwierig ist, andererseits war die Unterstützung für JUnit und JavaFX zum Zeitpunkt der Bachelorarbeit noch nicht weit ausgereift. Daher wurde die Benutzerschnittstelle durch interaktive, manuelle Tests geprüft.

Die Source-Ordner wurden nach Maven-Projektstruktur angelegt:
- src/main/java: sämtlicher Programmcode
- src/main/test: sämtlicher Testcode in paralleler Paket-Hierarchie aufgebaut

3.1.1 Emulator

Ein generelles Problem beim Test des API-Code war die Abhängigkeit von Hardware. Damit die Funktionalität trotzdem getestet werden konnte, wurden einerseits Klassen mit Testdaten implementiert (mehr dazu siehe 3.2 Interaktiv), andererseits wurde ein Emulator entwickelt, der für die Simulation von Kartenbefehlen benutzt werden kann.

Da sämtliche Hardware-Zugriffe über die *CardConnector*-Schnittstelle laufen, konnte ein Emulator entwickelt werden, der sich im *Learning*-Mode als Decorator an den *CardConnecter* „andockt“ und sämtliche APDUs, connects / disconnects und sämtliche angeschlossenen Kartenlesegeräte protokolliert und in eine Datei serialisiert. Im *Replay* – Modus ersetzt der Emulator dann den CardConnector, fängt die von der aufrufenden Klasse gesendeten Befehle ab und gibt die in der Datei gespeicherte Antwort auf das Datenpaket zurück. Wird die Reihenfolge der Aufrufe verletzt, so wirft der Emulator eine Exception. Der Emulator ist nicht für die interaktive Benutzung gedacht, sondern zur Simulation von Abläufen in einer bestimmten Reihenfolge.

UnitTests konnten so mit angeschlossener Hardware entwickelt und dann mit Hilfe des Emulators ohne Hardwarezugriff ausgeführt werden. So können die Tests auch auf dem Build-Server ausgeführt werden.

Erst wurde für diese Funktionalität die *EasyMock*-Bibliothek in Betracht gezogen. Sie war jedoch für diese spezifische Funktionalität zu umfangreich und umständlich. Aus diesem Grund wurde auf EasyMock verzichtet und stattdessen selbst ein Emulator entwickelt.

Die *CardCommunicator*-Klasse selbst kann so nicht automatisch getestet werden. Diese ist jedoch auf das absolute Minimum beschränkt (7 Methoden, geringe Komplexität), daher ist die Wahrscheinlichkeit dass etwas schief geht sehr gering.

3.1.2 Klassen-Tests

Im Cardaccess-Layer wurden die einzelnen Klassen, wo Sinnvoll, durch individuelle Tests der einzelnen Methoden geprüft. Manche Klassen konnten allerdings nur im grösseren Zusammenhang getestet werden, z.B. die *LowLevelCardFactory*. Im Dataaccess-Layer wurden weniger Klassen-spezifische UnitTests geschrieben, da dieser Layer vor allem Datenklassen enthält, dessen Getter- und Setter-Methoden nicht getestet werden müssen. Die Konsistenz der Datenklassen wird im *Administrative*- bzw. *MedicalDataTest* überprüft, indem ein kompletter Satz an Testdaten nach XML serialisiert und dann das Schema des erzeugten XML mit der Spezifikation des BAG validiert wird.

3.1.3 Layer-übergreifende Tests

Im Dataaccess-Layer wurden Tests für sämtliche Funktionen der Interfaces geschrieben, die mit dem Emulator bis auf die niedrigste Stufe zugreifen.

3.2 Interaktiv

Trotz Emulator konnten nicht alle Hardware-Funktionen automatisiert getestet werden. Für solche Fälle wurden Commandline-Tools geschrieben, z.B. für das Testen der Hardware-Änderungserkennung. Diese Tools befinden sich im Paket ch.hsr.egk.api.cardaccess.commandline. Das Programm HardwareChangeDetectionInteractive z.B. kann verwendet werden für den Test der automatischen Hardware-Änderungserkennung.

Für das Hardwareunabhängige Testen des User Interfaces wurden Mock-Implementierungen der Interfaces des API erstellt. Diese Klassen fanden später den Weg in den allgemeinen Workflow der Applikation, da sie es erlauben, die möglichen Daten gemäss BAG Spezifikation aufzuzeigen. Diese Klassen werden *VirtualSwissInsuranceCard* bzw. *VirtualHealthProfessionalCard* genannt. Wenn die entsprechende Setting aktiviert ist, wird immer eine virtuelle Karte angezeigt, auch wenn physisch kein Kartenlesegerät bzw. keine SmartCard vorhanden ist.

3.3 Manuell

Die User Interface-Funktionen wurden in mehreren Testläufen manuell getestet. Einerseits mit virtuellen Karten, andererseits mit physisch angeschlossenen Karten und mehreren Lesegeräten. Gefundene Fehler wurden im Issue-Tracker eingetragen oder gleich korrigiert.

4 Weiterentwicklung

Im technischen Bericht wurden bereits einige Weiterentwicklungspunkte skizziert. In diesem Kapitel sollen diese noch konkretisiert werden. Die hier beschriebenen Vorgänge setzen vertiefte Kenntnisse von SmartCard APDU-Befehlen, Kenntnisse des eCH-0064 Standards und der Detailspezifikationen von Post/SwissSign und SASIS.

4.1 Entschlüsselung und Parsen der Zertifikate

Als Vorstufe für das Card-To-Card Authentifizierungsverfahren müssten die auf der Karte vorhandenen Zertifikate entschlüsselt und geparst werden. Entsprechende Funktionalität wurde bereits vorbereitet, konnte jedoch noch nicht fertig implementiert werden. Das Lesen der Zertifikate-Rohdaten funktioniert bereits. Hierfür kann die Methode readTransparentFile(fileId) in BasicISO7816SmartCard verwendet werden.

Diese Rohdaten sind speziell codiert: (genaueres siehe Post Detailspezifikation 3.2.1)

Ein Beispiel einer CVC Datenstruktur (ASN.1)

```
7f21                                   // tag cvc
    81d5                               // len cvc
        5f37                           // tag part 1
        8180                           // len part 1
            6a                         // padding
            03                         // cpi
            4348445350600109           // car
            0000000000000004348534648610109 // chr
            44462e4e6f7400             // cha
            2b0e03020f                 // oid
            1512310911                 // cisd
```

Die 5F37-Datenstruktur ist auf der SmartCard jedoch verschlüsselt abgelegt. Hier wurde das Verschlüsselungsverfahren RSA/SHA-1 mit Message-Recovery gemäss ISO 9796-2, DS1 Option 1 (T=1) eingesetzt. Streng genommen wird hier von „Signieren“ und nicht von „Verschlüsseln“ gesprochen. Das Message-Recovery-Verfahren zeichnet sich jedoch dadurch aus, dass auf Grund der Signatur die zu Grunde liegenden Daten wiederhergestellt werden können (eigentlich nicht die Idee von RSA Signaturen).

Dieses Verfahren müsste implementiert werden, um die Datenstruktur zu entschlüsseln. Das Entschlüsseln geschieht jeweils mit den Public Key des entsprechenden Root-Zertifikates.

Die entschlüsselten Daten müssen dann nach Post/Sasis Spezifikation geparst werden.

Java bietet von Haus aus keine Methoden für Message Recovery nach ISO 9796-2. Kurze Recherchen haben gezeigt, dass hierfür vermutlich der Security Provider von bouncycastle.org in Frage kommt.

4.2 Card-To-Card Authentifizierung

Dieses Verfahren wurde in den Detailspezifikation nur sehr vage beschrieben. In der Post-Spezifikation wurde z.B. nicht angegeben, dass bei der ersten Benutzung der Versichertenkarten jeweils das HPC Root-Zertifikat mit dem ManageSecurityEnvironment-Kommando auf die Karte aufgebracht werden muss.
Die Schritte der Authentifizierung wurden in der PostInsuranceCard-Klasse implementiert und ausführlich kommentiert.

Generell ist der Ablauf folgender:
PDC = Versichertenkarte
HPC = Leistungserbringerkarte
MSE = Manage security environment
PSO = Perform security operation

1. Zertifikate gegenseitig importieren:

1. Root HPC CVC von HPC lesen, entschlüsseln, parsen
2. HPC CVC von HPC lesen, entschlüsseln, parsen
3. CVC von PDC lesen
4. PDC CVC in HPC importieren, mit Root prüfen (mit MSE, PSO)
5. HPC CVC in PDC importieren, mit Root prüfen (mit MSE, PSO)

2. Authentifizierung gegenseitig:

1. HPC erzeugt Challenge
2. PDC signiert challenge mit PDC private key
3. HPC prüft challenge (external authenticate) mit PDC public key
   // nun in Gegenrichtung. Das ist die eigentliche eCH-0064 C2C
4. PDC erzeugt challenge
5. HPC signiert challenge mit seinem private key (details siehe Post spec.)
6. PDC verifiziert Signatur (external authenticate) mit public key der HPC

Ein funktionierendes C2C-Verfahren mit Post Test-Zertifikaten kann in einem detaillierten APDU-Trace vom Kartenhersteller (siehe Abgabe-CD) betrachtet werden. Hier sind auch einige E-Mails mit weiteren Test-Zertifikaten (z.B. verschiedene Rollen gemäss eCH-0064) zu finden.

Um medizinische Daten auf die Post-Testzertifikate schreiben zu können, muss mit einem Test-HPC-Zertifikat (signiert von ECH-Root) die HPC Authentifizierung simuliert werden. Hierzu muss das Signieren des von der InsuranceCard erzeugten 8 Bit random Challenge (eigentlich vom HPC Betriebssystem vorgenommen) softwareseitig durchgeführt werden. Dazu wird ein 128 Bit digital signature input zusammengestellt und mit dem private Key des HPC-Zertifikates signiert. Dieses signierte Datenpaket wird an die InsuranceCard mit dem ExternalAuthenticate-Kommando geschickt. Wenn dieses das Statuswort 90 00 zurückgibt, war die Authentifizierung erfolgreich. Das Senden des Kommandos und auch das Signieren ist bereits implementiert, hat aber noch nicht funktioniert. Bei Fragen hierzu kann mit Michael Doujak von SwissSign Kontakt aufgenommen werden, der die Frage dann an einen Kartenspezialisten weiterleitet.

Das gleiche Verfahren kann möglicherweise auch mit der SASIS Karte durchgeführt werden, sofern SASIS Test-Zertifikate erstellt hat. Das kann Dominic Allemann von SASIS eventuell weiterhelfen.

4.3 Medizinische Daten von TLV in Datenobjekte umwandeln und zurück

In eCH-0064 wurden die meisten medizinischen EFs als Record-basierte Dateien definiert. Hier entspricht z.B. eine Impfung einem Record im File EF.IMMD. Es wurde spezifiziert, dass diese Records TLV codiert sind, die genaue Definition „welches Tag für welchen Wert“ wurde nirgends spezifiziert, weder in den Post-Detailspezifikationen noch in denen von SASIS. Da vermutlich ausser für Tests noch nie medizinische Daten geschrieben worden sind, existieren diese Definitionen noch nicht. Sie könnten vermutlich neu definiert werden in Absprache mit Post und SASIS.
Die im API implementierte TLV-Collection kann auch ohne genaue Definition von Tags benutzt werden, da Werte auch über deren Index gesetzt und gelesen werden können. So lassen sich diese Funktionen unabhängig von oben genannten Definitionen implementieren. Es gilt dann einfach die Reihenfolge der Werte innerhalb des Records.

Wenn das low-level handling (byte[] to TLVCollection) erledigt ist, werden die Daten an den data access Layer übergeben. Auf diesem Layer müssten Converter implementiert werden, welche die TLVCollections in Datenobjekte umwandeln und zurück. Hier können die Converter für administrative Daten als Referenz herbeigezogen werden.

4.4 Medizinische Daten: Schreib-Prinzip überarbeiten

Da die meisten medizinischen Daten Record-basiert sind, sollte das Schreib-Prinzip im API überarbeitet werden. Gemäss momentanem API-Design werden z.B. nicht einzelne Impfdaten geschrieben, sondern eine Liste d.h. das EF müsste geleert und mit den neuen Datensätzen überschrieben werden. Dieses Prinzip würde mit einer Änderungserkennung in den Schreib-Methoden funktionieren, was ursprünglich so geplant war, aber wegen fehlenden CVCs nicht umgesetzt werden konnte.

Eine Alternative wäre, den entsprechenden Datenobjekten auf dem Data Access Layer Record-IDs als Attribute hinzuzufügen und diese jeweils in den Convertern mitzugeben. Beim Schreiben könnte dann jeweils die Record-Id angegeben werden, um einen spezifischen Datensatz zu ersetzen.

4.5 User Interface

Das User Interface des Demonstrators kann ebenfalls erweitert werden. Interessant wäre die Möglichkeit, die gesendeten Kartenkommandos im User Interface anzuzeigen. Hierfür müsste auf *cardaccess*-Stufe ein Decorator für *cardcommunicator* geschrieben werden, der die Commands protokolliert (ähnlich dem Debugging-Decorator). Im User Interface könnte ein Thread implementiert werden, der stets Änderungen des Protokolls anzeigt (Achtung: JavaFX-Multithreading- Spezialitäten beachten!).

4.6 Kompatibilität mit anderen Diensten, PKCS#11

Ein weiterer Weiterentwicklungspunkt wäre das Prüfen der parallelen Lauffähigkeit von API und anderen auf den selben SmartCards basierenden Diensten.

Für Verschlüsselungs- und Signaturdienste, die auf der SmartCard vorhandene Zertifikate benutzen, wird in der Regel eine PKCS#11-Implementierung benutzt. Für die HPC existiert eine solche in Form einer .dll. Für Versichertenkarte konnte keine fertige Implementierung gefunden werden, weshalb hier auf opensc oder ähnliche Bibliotheken zurückgegriffen werden muss.

Falls es Kompatibilitätsprobleme geben sollte auf Grund von parallelen Kartenzugriffen, wäre eine Implementierung des API über die PKCS#11 Schnittstelle eine Option. Hierbei muss aber erst geklärt werden, inwiefern sich die Dienste und Daten der Versichertenkarte überhaupt über die Schnittstelle nutzen lassen. Eine mehr versprechende Lösung wäre, lediglich die Zugriffe auf die HPC über PKCS#11 zu führen. Hier muss jedoch geklärt werden, wie zwischen den beiden herstellerspezifischen Zertifikaten unterschieden werden kann.

Allgemein, wenn PKCS#11 genutzt werden soll, ist die Applikation nicht mehr ohne Installation lauffähig, da erst ein PKCS#11 Provider im System registriert werden muss.

5 Anhang

5.1 Abbildungsverzeichnis

5.2 Tabellenverzeichnis

5.3 Literaturverzeichnis

Elektronische Versichertenkarte

Statistiken

Status	Freigegeben
Klassifikation	Intern
Erstellt	10.06.10
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Dieses Dokument enthält statistische Daten zum Quellcode des Demonstrators.

1.2 Zweck

Verschaffen eines Überblicks über Umfang und Qualität des Quellcodes.

1.3 Gültigkeit

Gültig während der gesamten Projektdauer.

2 Statistiken

Die Statistiken wurden mit Sonar (www.sonarsource.com) erstellt. Sonar wurde in den Build-Prozess integriert, um einen kontinuierlichen Verlauf der Code-Qualität und des Umfangs zu erhalten.

Generell ist zu diesen Daten zu bemerken, dass die Klassen im Paket *ch.hsr.egk.api.dataaccess.healthcards.insurance.data* aus einem XML-Schema generiert worden sind. Die hohe Zahl an Duplications bezieht sich hauptsächlich auf diese generierten Klassen. Die Funktionalität und Integrität dieser generierten Klassen wurde mit verschiedenen Unit Tests verifiziert.

2.1 Gesamtprojekt

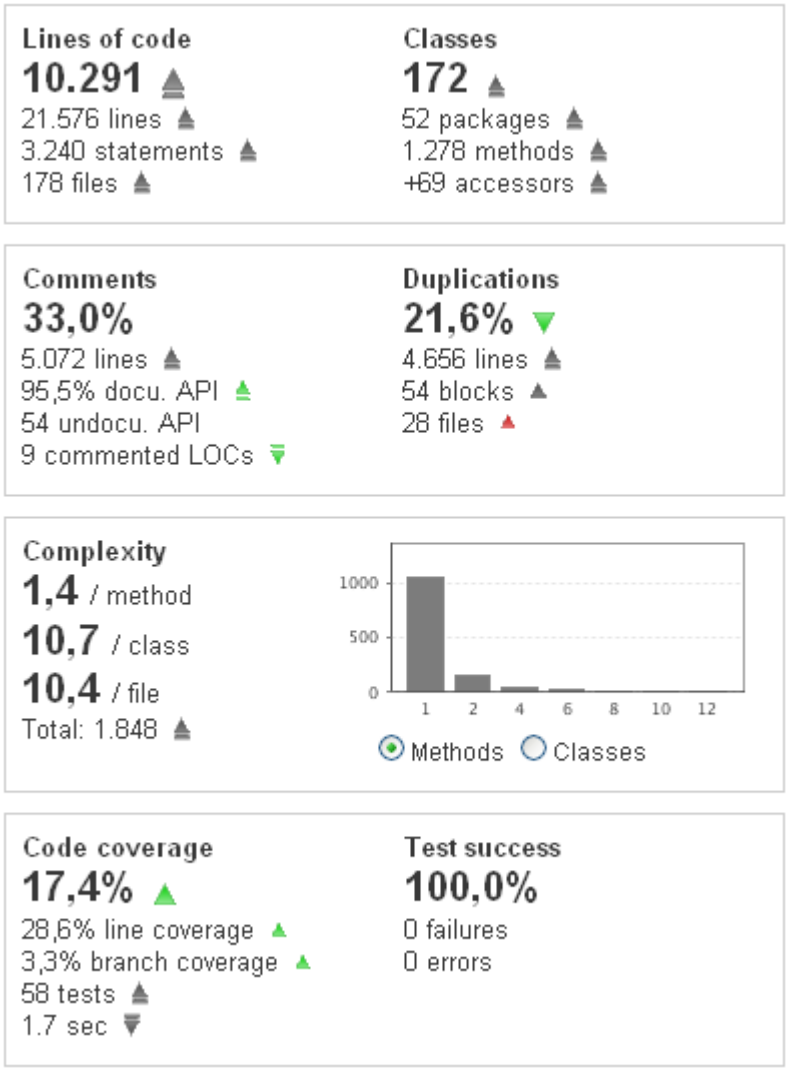


Abbildung 1: Statistiken Gesamtprojekt

2.2 API

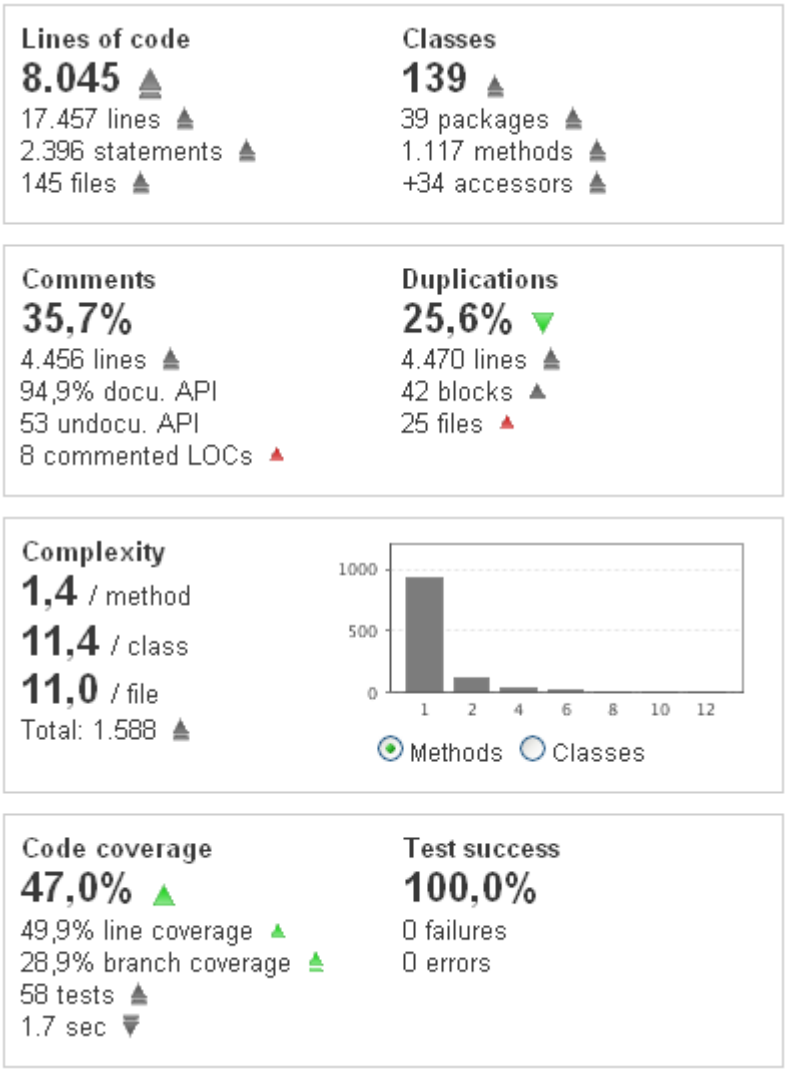


Abbildung 2: Statistiken API

2.3 User Interface

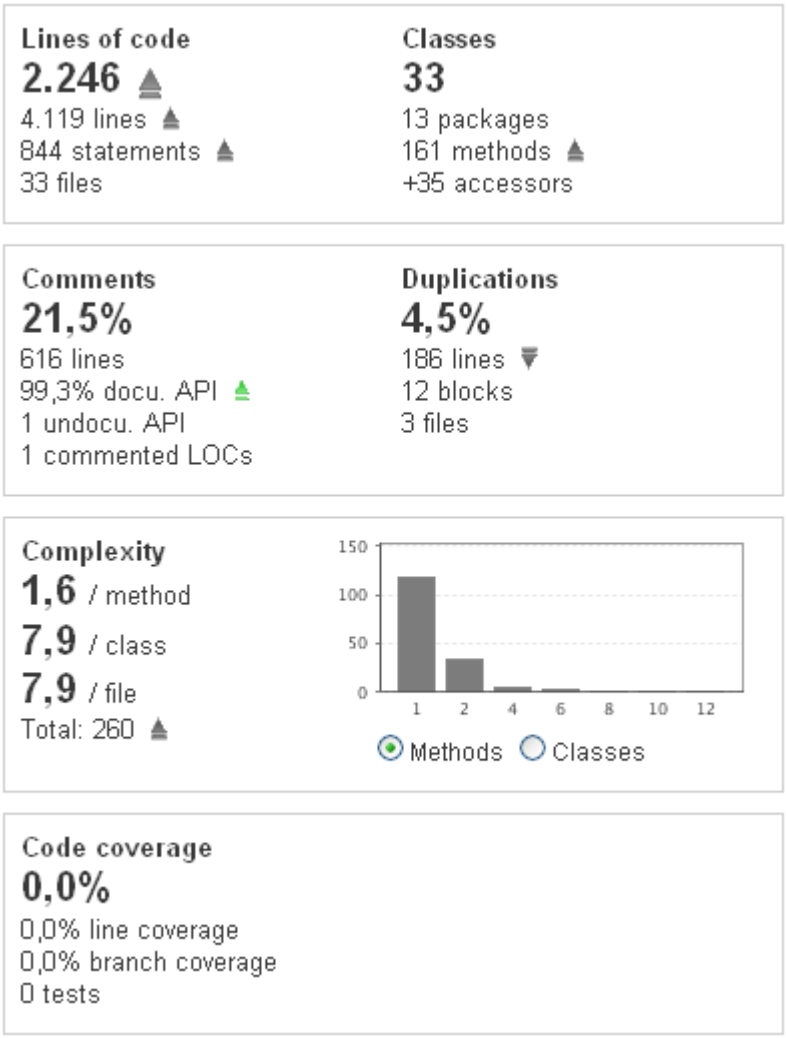


Abbildung 3: Statistiken User Interface

2.3.1 JavaFX-Quellcode

*.fx Dateien sind in obiger Statistik nicht enthalten, da sie in der aktuellen Version von Sonar nicht unterstützt werden. Folgende Werte wurden durch ein dafür geschriebenes Shell Script ermittelt:

Grösse	*Wert*
Quelldateien	33
Lines of code	3693
Kommentarzeilen	726
Kommentar %	19,6%

Table 1: Statistiken zu JavaFX Source

3 Anhang

3.1 Abbildungsverzeichnis

3.2 Tabellenverzeichnis

3.3 Literaturverzeichnis

Elektronische Versichertenkarte

Installation

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-06-10
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Client-Installation

Generell läuft der Demonstrator bei installiertem JRE > 1.6 ohne Installation über WebStart. Die nötigen JavaFX Runtime-Komponenten werden automatisch heruntergeladen.

Diese können aber auch manuell installiert werden und der Demonstrator kann dann auch direkt über sein JAR-File gestartet werden:

javafx -jar SwissInsuranceCardDemonstrator.jar

Wichtige Punkte, die es beim Client zu beachten gibt:

- Wenn etwas schief läuft, Java WebStart Cache leeren in Systemsteuerung > Java > Temporäre Internetdateien
- Ist der Pfad innerhalb der JNLP-Datei korrekt? (kann mit normalem Text-Editor kontrolliert werden)

Hardware:
Beim Client sollte ein Kartenlesegerät mit PC/SC-kompatiblem Treiber installiert sein. Es sollten ausserdem nach Möglichkeit keine weiteren Dienste in Betrieb sein, die parallel dazu auf SmartCards zugreifen (z.B. Crypto-Tokens usw.)

2 Installation Entwicklungsumgebung lokal (lokale Builds)

Für die Installation der Entwicklungsumgebung sind folgende Schritte nötig:

1. Installation Eclipse und JDK >= 1.6

2. Installation JavaFX 1.3 SDK
http://javafx.com/

3. JavaFX Eclipse Plugin
http://javafx.com/docs/gettingstarted/eclipse-plugin/

4. Import der beiden Projekte in Eclipse
- SwissInsuranceCardAPI
- SwissInsuranceCardUI

Das SwissInsuranceCardUI-Projekt muss eine Referenz auf das API-Projekt gesetzt haben.

5. Bibliotheken
Es müssen zudem folgende Bibliotheken im Classpath enthalten sein:
- jaxb-api.jpr
- jaxb-impl.jpr
- jaxb_20091104.jar (je nach Version, auch anderer Name)
- xercesImpl.jpr

Diese können hier heruntergeladen werden:
https://jaxb.dev.java.net/

Sowie hier:
http://xerces.apache.org/xerces-j/

Für das Ausführen von UnitTests wird zudem JUnit 4 benötigt.

Hinweis 1: Beim SwissInsuranceCardUI-Projekt müssen externe JARs direkt über ihren Pfad referenziert werden. Das Benutzen von Variablen funktioniert mit dem JavaFX Compiler nicht richtig.

Hinweis 2: Wenn noch nie mit JavaFX gearbeitet wurde, lohnt es sich, erst ein einfaches Testprojekt anzulegen, um sicher zu stellen, dass dieses korrekt ausgeführt werden kann.

Hinweis 3: Manchmal treten auch Probleme auf, weil JavaFX-Projekte nicht sauber gecleant werden. Deshalb kann es manchmal helfen, wenn mal etwas nicht richtig funktioniert, das ganze Projekt zu cleanen und neu zu builden.

6. Launch
Nun kann im SwissInsuranceCardUI-Projekt mit Rechtsklick auf *Main.fx*
das Programm gestartet werden. -> run as -> JavaFX Application

3 Installation für Maven

Damit das Projekt mit Maven kompiliert werden kann, werden einigen Maven-Plugins benötigt. Diese werden jedoch automatisch installiert, da sie in den jeweiligen pom.xml Files angegeben sind.

Auf dem Build-Server muss die JavaFX-SDK installiert sein und der Pfad zu dessen Binaries muss im pom.xml des SwissInsuranceCardUI-Projektes angegeben werden. Dieser kann im Abschnitt „configuration“ des Plugins *javafxc-maven-plugin* angepasst werden:

```
<javaFxHome>/svnroot/bin/javafx-sdk1.3/</javaFxHome>
```

Die Pfade auf die Dependencies müssen im SwissInsuranceCardUI-Projekt ebenfalls angepasst werden. Hier muss noch erwähnt sein, dass Maven-Builds momentan noch keine lauffähige JavaFX-Applikation erzeugen können, da das javafxc-maven-plugin einen Bug enthält, der in Absprache mit den Entwicklern des Plugins bereits gemeldet wurde.

Im Pom.xml des UI-Projektes sind ferner Zugangsdaten zum lokalen Keystore hinterlegt. Diese müssten natürlich mit den eigenen Angaben ersetzt werden.

Build des gesamten Projektes kann nachher auf SwissInsuranceCardDemonstrator-Stufe (unter den beiden Projekten) gestartet werden mit folgendem Command:

mvn clean install

4 Konfiguration Demonstrator

Der Demonstrator verfügt über verschiedene Konfigurationsmöglichkeiten, die nachfolgend aufgelistet sind.

4.1 ch.hsr.egk.presentation.preferences.Configuration

In dieser Datei können mit statischen Klassenvariablen, Einstellungen zu Aussehen und Funktionsweise des User Interfaces gemacht werden. Nachfolgend einige der nennenswertesten:

addVirtualHealthInsuranceCard	Legt fest, ob standardmässig eine virtuelle Versichertenkarte angezeigt werden soll. true = Standardmässig virtuelle Versichertenkarte anzeigen
addVirtualHealthProfessionalCard	Legt fest, ob standardmässig eine virtuelle HPC angezeigt werden soll. true = Standardmässig virtuelle HPC anzeigen
allowModifications	Aktiviert/Deaktiviert Editier-Funktionen und Save-Funktionalität freischalten (läuft aber solange keine CVCs für HPC ins leere). true = Änderungen der Daten ermöglichen
allowPinChange	Aktiviert den PIN-Änderungsdialog. true = PIN-Änderungsdialog freischalten
fullScreen	Legt fest, ob die Applikation im Full-Screen-Modus starten soll true = Applikation im Full-Screen-Modus starten

Table 1: Konfigurationsmöglichkeiten User Interface

4.2 ch.hsr.egk.presentation.language.LanguageManager

In dieser Datei können Einstellungen zu Sprachen und zu verwendeten Sprachdateien gemacht werden. Die Übersetzungen werden im untergeordneten Paket *properties* abgelegt.

currentLanguagePropertiesFile	Legt fest, ob standardmässig eine virtuelle Versichertenkarte angezeigt werden soll.
defaultLanguage	Legt fest, ob standardmässig eine virtuelle HPC angezeigt werden soll.
defaultCountry	Aktiviert/Deaktiviert Editier-Funktionen

Table 2: Konfigurationsmöglichkeiten Sprachen

4.3 ch.hsr.egk.api.misc.Settings

In dieser Datei können Einstellungen zum API gemacht werden.

ADD_DEMO_INSURANCE_CARD	Legt fest, ob standardmässig eine virtuelle Versichertenkarte angezeigt werden soll.
ADD_DEMO_HPC	Legt fest, ob standardmässig eine virtuelle HPC angezeigt werden soll.
ADMINISTRATIVE_DEMO_DATA_PATH	Gibt den Pfad zu einer XML-Datei Datei an, welche administrative Beispieldaten enthält. Achtung: die Datei muss vorher auf Schema-Validität geprüft werden.
MEDICAL_DEMO_DATA_PATH	Gibt den Pfad zu einer XML-Datei Datei an, welche medizinische Beispieldaten enthält. Achtung: die Datei muss vorher auf Schema-Validität geprüft werden.
LOG_LEVEL	Hier können verschiedene Log-Levels eingerichtet werden.
ENABLE_CARD_TRACE	Diese Option aktiviert die Debug-Ausgabe aller abgesetzten Kartenbefehle und deren Antworten.

Table 3: Konfigurationsmöglichkeiten API

5 Anhang

5.1 Abbildungsverzeichnis

5.2 Tabellenverzeichnis

5.3 Literaturverzeichnis

Elektronische Versichertenkarte

Projektplan

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-03-04
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Dieses Dokument soll die Organisation der Arbeit festlegen. Es umfasst oder referenziert die gesamte Planung der Arbeit.

1.2 Zweck

Es soll eine einwandfreie Durchführung garantiert und die Aufteilung der zu erledigenden Arbeiten festgelegt werden.

1.3 Gültigkeit

Dieses Dokument dient als Grundlage für die ganze Arbeit und hat deshalb Gültigkeit über die gesamte Dauer des Projekts. Änderungen werden laufend ergänzt und sind durch die Versionskontrolle ersichtlich.

2 Übersicht

2.1 Zweck und Ziel

Siehe 01 Diverses/Aufgabenstellung.

2.2 Annahmen und Einschränkungen

Pro Teammitglied wird eine durchschnittliche Arbeitszeit von ca. 360 Stunden insgesamt und ca. 21 Stunden pro Woche, über die Dauer des ganzen Projekts, erwartet. Falls nötig können die Zeiten erhöht werden.

3 Organisation

3.1 Organisationsstruktur

Unser Team besteht aus 2 Personen. Aufgrund der überblickbaren Teamgrösse verzichten wir auf einen Projektleiter.

Name	*Aufgabe*
Hofmann Michael mhofmann@hsr.ch	Projektausführend
Lowe Mark mlowe@hsr.ch	Projektausführend

Table 1: Projektausführende Personen

3.2 Externe Schnittstellen

Name	*Aufgabe*
Doering Axel 055 222 46 43 adoering@hsr.ch	Verantwortlicher, Betreuer

Table 2: Betreuende Personen

4 Management Abläufe

4.1 Zeitplan

Siehe 02 Zeitplanung.

4.2 Besprechungen

4.2.1 Teamsitzung

Tag	*Zeitslot*	*Ort*	*Personen*	*Weiteres*
Donnerstag	10:00-12:00	1.262	Hofmann Michael, Lowe Mark	**Teamsitzung** Review der gemachten Arbeiten. Detailplanung kommende Woche.

Table 3: Termine Besprechungen

4.2.2 Besprechung mit Betreuer

Mit dem Betreuer finden keine regelmässigen Besprechungen statt. Es können jedoch jederzeit Besprechungen angesetzt werden, falls erforderlich.

4.3 Abgaben

Zu jeder Arbeitswoche wird jeweils ein Wochenbericht verfasst. Bei fertigen Dokumenten erfolgt jeweils eine Benachrichtigung per Mail an alle Projektbeteiligten.

4.3.1 Wochenbericht

Die Wochenberichte sollen einen Überblick über den Projektverlauf bieten und die Projektarbeiten dokumentieren.

5 Risiko Management

RI-1	*Ausfall Teammitglied (< 3 Wochen)*
Auswirkungen:	• Verzögerungen
Massnahmen:	• Reserve einplanen
Schaden maximal:	45 Stunden
Wahrscheinlichkeit:	10 %
Priorität:	mittel

Table 4: Risiko Ausfall Teammitglied (< 3 Wochen)

RI-2	*Ausfall Teammitglied (> 3 Wochen)*
Auswirkungen:	• Verzögerungen
Massnahmen:	• Reserve einplanen, Projektumfang reduzieren
Schaden maximal:	120 Stunden
Wahrscheinlichkeit:	5 %
Priorität:	niedrig

Table 5: Risiko Ausfall Teammitglied (> 3 Wochen)

RI-3	*Datenverlust*
Auswirkungen:	• Teile des Projekts gehen verloren
Massnahmen:	• Versionskontrollsystem, Backups
Schaden maximal:	720 Stunden
Wahrscheinlichkeit:	3 %
Priorität:	mittel

Table 6: Risiko Datenverlust

RI-4	*Änderung Projektanforderungen*
Auswirkunge	• Ergänzung / Änderung der Software und Dokumentation • Überarbeitung der Projektplanung
Massnahmen:	• Flexible Software-Architektur, damit Erweiterungen möglich sind • Reserve einplanen
Schaden maximal:	80 Stunden
Wahrscheinlichkeit:	25 %
Priorität:	mittel

Table 7: Risiko Änderung Projektanforderungen

RI-5	*Plattformprobleme*
Auswirkungen:	• Applikation nicht mit allen geforderten Plattformen kompatibel
Massnahmen:	• Fortlaufende Plattformtests während Entwicklung
Schaden maximal:	10 Stunden
Wahrscheinlichkeit:	10 %
Priorität:	niedrig

Table 8: Risiko Plattformprobleme

RI-6	*Fehlerhafte Zeitplanung*
Auswirkungen:	• Nichteinhalten der Zeitvorgaben
Massnahmen:	• Reserven einplanen
Schaden maximal:	40 Stunden
Wahrscheinlichkeit:	20 %
Priorität:	hoch

Table 9: Risiko Fehlerhafte Zeitplanung

RI-7	*Kommunikationsprobleme*
Auswirkungen:	• Verzögerungen durch Mehraufwand
Massnahmen:	• Regelmässige Meetings • Alle E-Mails CC an andere Projektmitglieder • Wöchentlicher Bericht • Sitzungsprotokolle
Schaden maximal:	15 Stunden
Wahrscheinlichkeit:	10 %
Priorität:	mittel

Table 10: Risiko Kommunikationsprobleme

RI-8	*Fehlerhafte Hardware*
Auswirkungen:	• Nichteinhalten der Zeitvorgaben • Beschaffung Ersatzhardware
Massnahmen:	• Umfassende Evaluation • Frühe Beschaffung der Hardware
Schaden maximal:	80 Stunden
Wahrscheinlichkeit:	20 %
Priorität:	hoch

Table 11: Risiko Fehlerhafte Hardware

RI-9	*Falsche Hardware*
Auswirkungen:	• Verzögerungen durch Mehraufwand • Hardware nicht kompatibel mit eGK • Erneute Evaluation • Bestellung neuer Hardware • Zum API inkompatible Schnittstelle
Massnahmen:	• Umfassende Evaluation
Schaden maximal:	100 Stunden
Wahrscheinlichkeit:	30 %
Priorität:	hoch

Table 12: Risiko Falsche Hardware

RI-10	*Verzögerungen Beschaffung Hardware*
Auswirkungen:	• Verzögerung durch warten auf zu beschaffende Hardware • Beschaffung alternativer Hardware mit erneuter Evaluation
Massnahmen:	• Bei Evaluation auf Lieferfristen achten • Notfalls Beschaffung Ersatz
Schaden maximal:	8 Stunden
Wahrscheinlichkeit:	10 %
Priorität:	hoch

Table 13: Risiko Falsche Hardware

RI-11	*Unvollständige Dokumentation der Standards (eGK)*
Auswirkungen:	• Nichteinhalten der Zeitvorgaben • Änderung Projektanforderungen • Beschaffung falscher Hardware
Massnahmen:	• Kritische Betrachtung Standards / Spezifikationen
Schaden maximal:	120 Stunden
Wahrscheinlichkeit:	25 %
Priorität:	hoch

Table 14: Unvollständige / fehlerhafte Dokumentation des Standards (eGK)

RI-12	*Fehlerhafte Dokumentation der Standards (eGK)*
Auswirkungen:	• Nichteinhalten der Zeitvorgaben • Änderung Projektanforderungen • Beschaffung falscher Hardware
Massnahmen:	• Kritische Betrachtung Standards / Spezifikationen
Schaden maximal:	120 Stunden
Wahrscheinlichkeit:	5 %
Priorität:	hoch

Table 15: Unvollständige / fehlerhafte Dokumentation des Standards (eGK)

RI-13	*Änderung des Standards durch politische Einflüsse*
Auswirkungen:	• Nichteinhalten der Zeitvorgaben • Änderung Projektanforderungen • Änderungen an Hardwareanforderungen (→ falsche Hardware beschafft)
Massnahmen:	• Beobachten der politischen Entwicklung (z.B. auf Websites BAG, Santésuisse, Bund)
Schaden maximal:	120 Stunden
Wahrscheinlichkeit:	10 %
Priorität:	niedrig

Table 16: Änderung des Standards durch politische Einflüsse (eGK)

RI-14	*Entwickeltes System nicht kompatibel mit realen Gesundheitskarten*
Auswirkungen:	• Wertsenkung der Demonstrator-Applikation • Anpassungen an Applikation

Massnahmen:	• Test wenn möglich mit realen Gesundheitskarten
	  • Anforderung Testkarten bei zuständiger Stelle
	  • Test mit eigenen Karten
	• Genaue Einhaltung Spezifikationen

Schaden maximal:	50 Stunden
Wahrscheinlichkeit:	20 %
Priorität:	mittel

Table 17: Unvollständige / fehlerhafte Dokumentation des Standards (eGK)

RI-15	*Kein Zugriff auf geschützte Daten der Gesundheitskarte*
Auswirkungen:	• Wertsenkung der Demonstrator-Applikation • Demo nur mit Testkarten möglich
Massnahmen:	• Abklären, ob Daten auch ohne Leistungserbringer- Authentifizierung gelesen werden können • Allenfalls Anfordern einer Leistungserbringer-Authentifizierung (testweise) bei den zuständigen Stellen
Schaden maximal:	20 Stunden
Wahrscheinlichkeit:	30 %
Priorität:	mittel

Table 18: Kein Zugriff auf medizinische Daten der Gesundheitskarte

5.1.1 Neubeurteilung in Projektwoche 7

In Projektwoche 7 wurden die Risiken neu beurteilt. Wichtigstes Risiko ist neu:

RI-16	*Keine Lieferung von signierten CVCs durch FMH*
Auswirkungen:	• Wertsenkung der Demonstrator-Applikation • Demo nur mit administrativen Daten möglich
Massnahmen:	• Abklären, wie und ob Post-Testzertifikate verwendet werden können • Virtualisierung der medizinischen Daten, damit deren Zusammensetzung vom User Interface trotzdem angezeigt werden kann
Schaden maximal:	20 Stunden
Wahrscheinlichkeit:	60.00%
Priorität:	sehr hoch

Table 19: Keine Lieferung von CVCs durch FMH

5.2 Massnahmen

5.2.1 Neubeurteilung der Risiken

Nach erreichen jedes Meilensteins werden die Risiken durch ein Projektmitglied überprüft und neu beurteilt.

5.2.2 Reserve

Aufgrund obiger Berechnung planen wir 2 Wochen als Reserve ein.

5.2.3 Spezifische Risiken frühzeitig erkennen

Genaues Studium der Standards, der Hardware und der Software.

5.2.4 Probleme frühzeitig erkennen

Um die Auswirkungen von Problemen möglichst gering zu halten, werden kritische Arbeiten zu einem möglichst frühen Zeitpunkt erledigt oder geprüft (falls Erledigung erst zu einem späteren Zeitpunkt möglich).

6 Arbeitspakete

Der in den Arbeitspaketen angegebene Aufwand entspricht nur Schätzwerten zur Grobplanung und kann in der Detailplanung jeweils am Ende jeder Phase angepasst werden. Siehe Dokumente 02 Zeitplan.

6.1 Projektplan

AP-1-1	*Projektorganisation*
Arbeitsaufwand:	1 Stunde
Beschreibung:	• Erstellen der Organisationsstruktur • Dokumentieren der externen Schnittstellen
Abhängigkeiten:	-
Risiken:	• RI-7: Kommunikationsprobleme
Artefakte:	Projektplan

Table 20: Arbeitspaket Projektorganisation

AP-1-2	*Management Abläufe*
Arbeitsaufwand:	2 Stunden
Beschreibung:	• Erstellen der groben Zeitplanung • Festlegen der Meilensteine (inklusive Termine) • Planung der Besprechungen
Abhängigkeiten:	-
Risiken:	• RI-6: Fehlerhafte Zeitplanung • RI-7: Kommunikationsprobleme
Artefakte:	Projektplan

Table 21: Arbeitspaket Management Abläufe

AP-1-3	*Risiko Management*
Arbeitsaufwand:	2 Stunden
Beschreibung:	• Mögliche Risiken erkennen • Deren Auswirkungen herausfinden • Massnahmen entwickeln • Schaden abschätzen
Abhängigkeiten:	-
Risiken:	-
Artefakte:	Projektplan

Table 22: Arbeitspaket Risiko Management

AP-1-4	*Arbeitspakete*
Arbeitsaufwand:	3 Stunden
Beschreibung:	• Arbeitspakete definieren • Arbeitsaufwand abschätzen • Arbeitsergebnisse festlegen • Risiken erkennen
Abhängigkeiten:	-
Risiken:	• RI-6: Fehlerhafte Zeitplanung
Artefakte:	Projektplan

Table 23: Arbeitspaket Arbeitspakete

AP-1-5	*Infrastruktur*
Arbeitsaufwand:	1 Stunde
Beschreibung:	• Benötigte Infrastruktur dokumentieren
Abhängigkeiten:	-
Risiken:	-
Artefakte:	Projektplan

Table 24: Arbeitspaket Infrastruktur

AP-1-6	*Hardware*
Arbeitsaufwand:	3 Stunden
Beschreibung:	• Benötigte Hardware evaluieren • Benötigte Hardware beschaffen
Abhängigkeiten:	-
Risiken:	• RI-8: Fehlerhafte Hardware • RI-9: Falsche Hardware
Artefakte:	Hardware

Table 25: Arbeitspaket Hardware

AP-1-7	*Qualitätsmassnahmen*
Arbeitsaufwand:	1 Stunde
Beschreibung:	• Zu verwendende Qualitätsmassnahmen festlegen
Abhängigkeiten:	-
Risiken:	-
Artefakte:	Projektplan

Table 26: Arbeitspaket Qualitätsmassnahmen

6.2 Zeitplan

AP-2	*Zeitplan*
Arbeitsaufwand:	3 Stunden
Beschreibung:	• Planung der Abarbeitung der einzelnen Arbeitspakete
Abhängigkeiten:	• AP-1-4: Arbeitspakete
Risiken:	• RI-1: Ausfall Teammitglied (< 3 Wochen) • RI-2: Ausfall Teammitglied (> 3 Wochen) • RI-6: Fehlerhafte Zeitplanung
Artefakte:	Zeitplan

Table 27: Arbeitspaket Zeitplan

6.3 Dokumentation

6.3.1 Recherche

AP-3-1-1	*Recherche Standards und Referenzimplementationen*
Arbeitsaufwand:	40 Stunden
Beschreibung:	• Recherche Standards und Referenzimplementationen
Abhängigkeiten:	• AP-1-6: Hardware
Risiken:	• RI-3: Datenverlust • RI-4: Änderung Projektanforderungen
Artefakte:	Dokumentation für die Lehre

Table 28: Arbeitspaket Recherche Standards und Referenzimplementationen

AP-3-1-2	*Einarbeitung SmartCard Grundlagen*
Arbeitsaufwand:	35 Stunden
Beschreibung:	• Recherche SmartCard Grundlagen
Abhängigkeiten:	• AP-1-6: Hardware
Risiken:	• RI-3: Datenverlust • RI-4: Änderung Projektanforderungen
Artefakte:	Dokumentation für die Lehre

Table 29: Arbeitspaket Einarbeitung SmartCard Grundlagen

6.3.2 Dokumentation für die Lehre

AP-3-2	*Dokumentation für die Lehre (Gesamtpaket)*
Arbeitsaufwand:	110 Stunden

Beschreibung:	• Übersicht (Entwicklungsstand der Versichertenkarte)
	  • Anbieter
	  • Standards
	  • Referenzimplementationen
	  • Features
	  • Infrastruktur
	  • Datenspeicherung
	  • Datenschutz
	  • Datenspeicherung
	  • Rechtliches
	  • Chancen und Risiken
	  • Alternativen
	• Technologie
	  • Hardware
	  • SmartCard
	  • CardReader
	  • Software
	  • APIs
	• Tutorial
	• Präsentation
	• Skript
	• Bedienungsanleitung Demonstrator

Abhängigkeiten:	• AP-1-6: Hardware
Risiken:	• RI-3: Datenverlust • RI-4: Änderung Projektanforderungen
Artefakte:	Dokumentation für die Lehre

Table 30: Arbeitspaket Dokumentation für die Lehre

AP-3-2-3	*Skript für die Lehre*
Arbeitsaufwand:	10 Stunden
Beschreibung:	• Erstellung Skript für die Lehre
Abhängigkeiten:	• AP-1-6: Hardware
Risiken:	• RI-3: Datenverlust • RI-4: Änderung Projektanforderungen
Artefakte:	Dokumentation für die Lehre

Table 31: Arbeitspaket Skript für die Lehre

AP-3-2-4	*Tutorial für die Lehre*
Arbeitsaufwand:	5 Stunden
Beschreibung:	• Erstellung Skript für die Lehre
Abhängigkeiten:	• AP-1-6: Hardware
Risiken:	• RI-3: Datenverlust • RI-4: Änderung Projektanforderungen
Artefakte:	Dokumentation für die Lehre

Table 32: Arbeitspaket Tutorial für die Lehre

AP-3-2-5	*Präsentation für die Lehre*
Arbeitsaufwand:	10 Stunden
Beschreibung:	• Erstellung Skript für die Lehre
Abhängigkeiten:	• AP-1-6: Hardware
Risiken:	• RI-3: Datenverlust • RI-4: Änderung Projektanforderungen
Artefakte:	Dokumentation für die Lehre

Table 33: Arbeitspaket Präsentation für die Lehre

AP-3-2-6	*Bedienungsanleitung Demonstrator*
Arbeitsaufwand:	10 Stunden
Beschreibung:	• Erstellung Skript für die Lehre
Abhängigkeiten:	• AP-1-6: Hardware
Risiken:	• RI-3: Datenverlust • RI-4: Änderung Projektanforderungen
Artefakte:	Dokumentation für die Lehre

Table 34: Arbeitspaket Bedienungsanleitung Demonstrator

6.3.3 Dokumentation der Arbeit

AP-3-3-1	*Dokumentation der Arbeit*
Arbeitsaufwand:	40 Stunden
Beschreibung:	• Titelblatt • Aufgabenstellung • Erklärung über die eigenständige Arbeit • Abstract / Kurzfassung • Kurzbeschreibung Diplombroschüre • Poster • Technischer Bericht • Persönliche Berichte • Glossar • Literaturverzeichnis • Dokumention des Projekts
Abhängigkeiten:	• AP-3-2: Dokumentation für die Lehre
Risiken:	• RI-3: Datenverlust
Artefakte:	Dokumentation der Arbeit

Table 35: Arbeitspaket Dokumentation der Arbeit

AP-3-3-2	*Veröffentlichung für eprints.hsr.ch*
Arbeitsaufwand:	20 Stunden
Beschreibung:	• Veröffentlichungsdokument erstellen
Abhängigkeiten:	• AP-3-2: Dokumentation für die Lehre • AP-3-2: Dokumentation der Arbeit
Risiken:	• RI-3: Datenverlust
Artefakte:	Veröffentlichungsdokument

Table 36: Arbeitspaket Dokumentation der Arbeit

6.4 Demonstrator

6.4.1 Anforderungen

AP-4-1-1	*Funktionale Anforderungen*
Arbeitsaufwand:	6 Stunden
Beschreibung:	• Festlegen der Funktionalen Anforderungen
Abhängigkeiten:	-
Risiken:	• RI-4: Änderung Projektanforderungen
Artefakte:	Anforderungsdokument

Table 37: Arbeitspaket Funktionale Anforderungen

AP-4-1-2	*Nichtfunktionale Anforderungen*
Arbeitsaufwand:	6 Stunden

Beschreibung:	• Festlegen der Nichtfunktionalen Anforderungen
	  • Leistungsanfoderungen
	  • Mengenanforderungen
	  • Schnittstellenanforderungen
	  • Randbedingungen
	  • Qualitätsmerkmale
	  • Weitere Anforderungen

Abhängigkeiten:	-
Risiken:	• RI-4: Änderung Projektanforderungen
Artefakte:	Anforderungsdokument

Table 38: Arbeitspaket Nichtfunktionale Anforderungen

AP-4-1-3	*Use Cases*
Arbeitsaufwand:	8 Stunden
Beschreibung:	• Use Cases erarbeiten • Aktoren herausfinden • Prioritäten festlegen • Use Cases hoher Priorität Fully Dressed dokumentieren • Use Cases niedrigerer Priorität Casual dokumentieren
Abhängigkeiten:	• AP-4-1-1: Funktionale Anforderungen • AP-4-1-2: Nichtfunktionale Anforderungen
Risiken:	• RI-4: Änderung Projektanforderungen
Artefakte:	Anforderungsdokument, Use Case Diagramm

Table 39: Arbeitspaket Use Cases

6.4.2 Analyse

AP-4-2-1	*Domain Modell*
Arbeitsaufwand:	10 Stunden
Beschreibung:	• Erstellen des Domainmodells • Beschreiben der verwendeten Konzepte
Abhängigkeiten:	• AP-4-1: Anforderungen
Risiken:	• RI-4: Änderung Projektanforderungen
Artefakte:	Analysedokument, Domain Modell

Table 40: Arbeitspaket Domain Modell

AP-4-2-2	*Systemsequenzdiagramme*
Arbeitsaufwand:	20 Stunden
Beschreibung:	• Erstellen von Sequenzdiagrammen für die entsprechenden Use Cases
Abhängigkeiten:	• AP-4-1: Anforderungen • AP-4-2-1: Domain Modell
Risiken:	-
Artefakte:	Analysedokument, Systemsequenzdiagramme

Table 41: Arbeitspaket Systemsequenzdiagramme

AP-4-2-3	*Operation Contracts*
Arbeitsaufwand:	10 Stunden
Beschreibung:	• Aufsetzen der Operation Contracts für die entsprechenden Methoden
Abhängigkeiten:	• AP-4-2-2: Systemsequenzdiagramme
Risiken:	-
Artefakte:	Analysedokument

Table 42: Arbeitspaket Operation Contracts

6.4.3 Design

AP-4-3-1	*Architektur*
Arbeitsaufwand:	10 Stunden
Beschreibung:	• Architekturziele definieren • Einschränkungen definieren • Physische Architektur entwickeln • Logische Architektur entwickeln • Paketschnittstellen definieren • Architekturkonzepte dokumentieren
Abhängigkeiten:	• AP-4-2: Analyse
Risiken:	• RI-4: Änderung Projektanforderungen
Artefakte:	Designdokument

Table 43: Arbeitspaket Architektur

AP-4-3-2	*Reale Use Cases*
Arbeitsaufwand:	10 Stunden
Beschreibung:	• Use Cases Beschreiben • Interaktionsdiagramme für entsprechende Use Cases erstellen
Abhängigkeiten:	• AP-4-2: Analyse
Risiken:	• RI-4: Änderung Projektanforderungen
Artefakte:	Designdokument

Table 44: Arbeitspaket Reale Use Cases

AP-4-3-3	*Externes Design*
Arbeitsaufwand:	15 Stunden
Beschreibung:	• User Interface spezifizieren • Abläufe festlegen • Oberflächenentwürfe erstellen
Abhängigkeiten:	• AP-4-2: Analyse
Risiken:	• RI-4: Änderung Projektanforderungen
Artefakte:	Designdokument

Table 45: Arbeitspaket Externes Design

AP-4-3-4	*Datenmodell*
Arbeitsaufwand:	5 Stunden
Beschreibung:	• Datenmodell entwickeln • Datensicherungsmethode spezifizieren • Persistenzüberlegungen anstellen
Abhängigkeiten:	• AP-4-2: Analyse • AP-4-3-1: Architektur
Risiken:	• RI-4: Änderung Projektanforderungen
Artefakte:	Designdokument

Table 46: Arbeitspaket Datenmodell

AP-4-3-5	*Technologien*
Arbeitsaufwand:	20 Stunden
Beschreibung:	• Verwendete Technologien dokumentieren
Abhängigkeiten:	• AP-4-2: Analyse • AP-4-3: Design (ausser sich selbst)
Risiken:	• RI-5: Plattformprobleme • RI-8: Fehlerhafte Hardware • RI-9: Falsche Hardware
Artefakte:	Designdokument

Table 47: Arbeitspaket Technologien

6.4.4 Entwicklung

AP-4-4-1	*Prototyp (Construction Phase 1)*
Arbeitsaufwand:	25 Stunden
Beschreibung:	• Ersten lauffähigen Prototypen erstellen
Abhängigkeiten:	• AP-4-1: Anforderungen • AP-4-2: Analyse • AP-4-3: Design
Risiken:	• RI-3: Datenverlust • RI-7: Kommunikationprobleme
Artefakte:	Sourcecode

Table 48: Arbeitspaket Prototyp

AP-4-4-2	*Beta-Version (Construction Phase 2)*
Arbeitsaufwand:	30 Stunden
Beschreibung:	• Erstes funktionsfähiges Programm erstellen • Programm erweitern
Abhängigkeiten:	• AP-4-1: Anforderungen • AP-4-2: Analyse • AP-4-3: Design • AP-4-4-1: Prototyp
Risiken:	• RI-3: Datenverlust • RI-7: Kommunikationprobleme
Artefakte:	Sourcecode

Table 49: Arbeitspaket Beta-Version

AP-4-4-3	*Finale-Version (Construction Phase 3)*
Arbeitsaufwand:	100 Stunden
Beschreibung:	• Fertiges Programm erstellen • Letzte Anpassungen vornehmen
Abhängigkeiten:	• AP-4-1: Anforderungen • AP-4-2: Analyse • AP-4-3: Design • AP-4-4-1: Prototyp • AP-4-4-2: Beta-Version
Risiken:	• RI-3: Datenverlust • RI-7: Kommunikationprobleme • RI-8: Fehlerhafte Hardware • RI-9: Falsche Hardware
Artefakte:	Sourcecode

Table 50: Arbeitspaket Finale-Version

6.4.5 Dokumentation

AP-4-5-1	*Systemdokumentation*
Arbeitsaufwand:	20 Stunden

Beschreibung:	• Dokumentieren von:
	  • Wichtige Konzepte
	  • Benutzeroberfläche
	  • Festlegungen
	  • Use Cases
	  • Systemarchitektur
	  • Zustandsmodellierung
	  • Synchronisation

Abhängigkeiten:	• AP-4-3: Design • AP-4-4: Entwicklung
Risiken:	-
Artefakte:	Systemdokumentation

Table 51: Arbeitspaket Systemdokumentation

AP-4-5-2	*Benutzerhandbuch*
Arbeitsaufwand:	20 Stunden
Beschreibung:	• Benutzerhandbuch erstellen
Abhängigkeiten:	• AP-4-3: Design • AP-4-4: Entwicklung
Risiken:	-
Artefakte:	Benutzerhandbuch

Table 52: Arbeitspaket Benutzerhandbuch

AP-4-5-3	*Installationsanleitung*
Arbeitsaufwand:	6 Stunden
Beschreibung:	• Installationsschritte dokumentieren
Abhängigkeiten:	• AP-4-3: Design • AP-4-4: Entwicklung
Risiken:	-
Artefakte:	Installationsanleitung

Table 53: Arbeitspaket Installationsanleitung

AP-4-5-4	*Javadoc*
Arbeitsaufwand:	4 Stunden
Beschreibung:	• Javadoc erstellen
Abhängigkeiten:	• AP-4-4: Entwicklung
Risiken:	-
Artefakte:	Javadoc

Table 54: Arbeitspaket Javadoc

6.4.6 Testen

AP-4-6-1	*Unit Tests*
Arbeitsaufwand:	20 Stunden
Beschreibung:	• Unit Tests dokumentieren
Abhängigkeiten:	• AP-4-4: Entwicklung
Risiken:	-
Artefakte:	Testprotokoll

Table 55: Arbeitspaket Unit Tests

AP-4-6-2	*Funktionale Tests*
Arbeitsaufwand:	25 Stunden
Beschreibung:	• Testen des Programms auf Funktionalität • Testspezifikationen definieren • Testfälle festlegen • Dokumentieren der Tests
Abhängigkeiten:	• AP-4-4: Entwicklung
Risiken:	-
Artefakte:	Testprotokoll

Table 56: Arbeitspaket Funktionale Tests

AP-4-6-3	*Usability-Tests*
Arbeitsaufwand:	15 Stunden
Beschreibung:	• Testen des Programms durch Drittpersonen • Testspezifikationen definieren • Testfälle festlegen • Dokumentieren der Tests
Abhängigkeiten:	• AP-4-4: Entwicklung
Risiken:	-
Artefakte:	Testprotokoll

Table 57: Arbeitspaket Usability-Tests

6.5 Präsentation

AP-5-1	*Präsentation*
Arbeitsaufwand:	10 Stunden

Beschreibung:	• Ablauf Präsentation festlegen
	• Präsentation für Demonstrator erstellen
	• Testumgebung für Demonstrator erstellen
	  • Testdaten
	  • Hardware vorbereiten
	  • Software einrichten

Abhängigkeiten:	• AP-3: Dokumentation • AP-4: Demonstrator
Risiken:	-
Artefakte:	Präsentation

Table 58: Arbeitspaket Präsentation

AP-5-2	*Präsentieren*
Arbeitsaufwand:	4 Stunden
Beschreibung:	• Präsentationsumgebung einrichten • Präsentieren der Arbeit • Präsentation des Demonstrators
Abhängigkeiten:	• AP-5-1: Präsentation
Risiken:	-
Artefakte:	Präsentation

Table 59: Arbeitspaket Präsentieren

6.6 Reserve

AP-6	*Reserve*
Arbeitsaufwand:	60 Stunden
Beschreibung:	• Noch nicht erledigte Arbeiten erledigen • Implementation von Zusatzfeatures gemäss Aufgabenstellung
Abhängigkeiten:	• AP-3: Dokumentation • AP-4: Demonstrator • AP-5: Präsentation
Risiken:	-
Artefakte:	-

Table 60: Arbeitspaket Reserve

7 Infrastruktur

7.1 Hardware

- Private Notebooks
- Private Desktops
- Computer HSR
- Linux-Server HSR
- Drucker
  - Privat
  - HSR
- SmartCard
- CardReader

7.2 Software

7.2.1 Betriebssystemsoftware

- Microsoft Windows XP
- Canonical Linux Ubuntu

7.2.2 Anwendungssoftware

- Eclipse
- Enterprise Architect
- Violet UML Editor
- OpenOffice
- Microsoft Office
- Acrobat Reader
- Crimson Editor
- SVN
- Tortoise SVN
- Gimp
- Paint .NET
- Firefox

7.2.3 Entwickler-Software

Details zu den eingesetzten Entwicklungstools können im Implementierungs-Teil der Softwaredokumentation entnommen werden.

7.2.4 Issue- Tracking

Fürs Issue- Tracking während dem Entwicklungsprozess wurde Trac eingesetzt. Tasks und zu korrigierende Bugs wurden jeweils laufend erfasst und abgearbeitet.

7.3 Weitere

- Internetzugang
- WLAN
- Arbeitsräume HSR

8 Qualitätsmassnahmen

8.1 Dokumentation

Die Dokumentation wird nach jedem Arbeitsschritt, soweit erforderlich, entsprechend ergänzt. Ausserdem werden sämtliche Richtlinien innerhalb der Dokumentation festgelegt.

8.2 Reviews

Sämtliche Teammitglieder treffen sich einmal wöchentlich zu einem Review.

8.3 Versionskontrolle

Während der gesamten Entwicklung wird ein Versionskontrollsystem eingesetzt, was es ermöglicht, den gesamten Entwicklungsvorgang zu verfolgen und jegliche Änderungen rückgängig zu machen.

8.4 Backups

Zur Minimierung des Datenverlustrisikos, werden in periodischen Abständen Backups erstellt.

8.5 Softwareentwicklung

8.5.1 Unit-Tests

Wo es sinnvoll ist, werden Unit-Tests zur fortlaufenden Testung der Funktionalität erstellt. Alle Programme werden mittels Unit Tests automatisiert getestet. Nach Änderungen an den Quelldateien muss immer zuerst der Unit Test wieder erfolgreich ablaufen, bevor weitere Änderungen am Code vorgenommen werden. Weiter muss der Code vor dem Einchecken ins Versionskontrollsystem den Unit-Test, sofern vorhanden, erfolgreich bestehen.

8.5.2 Funktionale Tests

Unabhängig von den Unit-Tests muss jeder Code bevor er im Versionskontrollsystem eingecheckt wird, vom entsprechenden Entwickler selbständig auf Funktionalität und Fehlerfreiheit getestet werden.

8.5.3 Usability-Tests

Nach Erstellung der ersten Prototypen, sowie nach grösseren Änderungen, wird ein Usability Test, von den Entwicklern oder unabhängigen Testpersonen, durchgeführt, protokolliert und allfällige Verbesserungen vorgenommen.

8.5.4 Server-Builds

Die Software wird auf einem unabhängigen Betriebssystem kompiliert. Dies fördert die Plattformunabhängigkeit.

9 Anhang

9.1 Abbildungsverzeichnis

9.2 Tabellenverzeichnis

9.3 Literaturverzeichnis

Elektronische Versichertenkarte

Zeitplanung

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-03-07
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Dieses Dokument beschreibt die Zeitplanung des Projekts.

1.2 Zweck

Es soll eine klar geregelte Zeitplanung erstellt und dokumentiert werden.

1.3 Gültigkeit

Dieses Dokument dient als Grundlage für die ganze Arbeit und hat deshalb Gültigkeit über die gesamte Dauer des Projekts. Änderungen werden laufend ergänzt und sind durch die Versionskontrolle ersichtlich.

2 Zeitplan

2.1.1 Zu Projektbeginn

Gewisse Arbeiten können auch Nebenläufig erfolgen. Das folgende Diagramm legt lediglich die primär zu verrichtenden Tätigkeiten fest.

ID	Task Name	Start	Finish	Duration
1	AP-1: Projektplan	01.03.2010	05.03.2010	13h
2	AP-2: Zeitplan	06.03.2010	07.03.2010	3h
3	AP-3-1: Recherche	08.03.2010	18.03.2010	75h
4	AP-4-1: Anforderungen (Demonstrator)	19.03.2010	21.03.2010	20h
5	AP-4-2: Analyse (Demonstrator)	22.03.2010	27.03.2010	40h
6	AP-4-3: Design (Demonstrator)	28.03.2010	05.04.2010	60h
7	AP-4-4: Entwicklung (Demonstrator)	06.04.2010	28.04.2010	155h
8	AP-4-5: Dokumentation (Demonstrator)	29.04.2010	05.05.2010	50h
9	AP-4-6: Testen (Demonstrator)	06.05.2010	14.05.2010	60h
10	AP-3-3:Dokumentation der Arbeit	15.05.2010	23.05.2010	60h
11	AP-3-2: Dokumentation für die Lehre	24.05.2010	08.06.2010	110h
12	AP-6 Reserve	09.06.2010	16.06.2010	60h
13	AP-5 Präsentation	17.06.2010	18.06.2010	14h

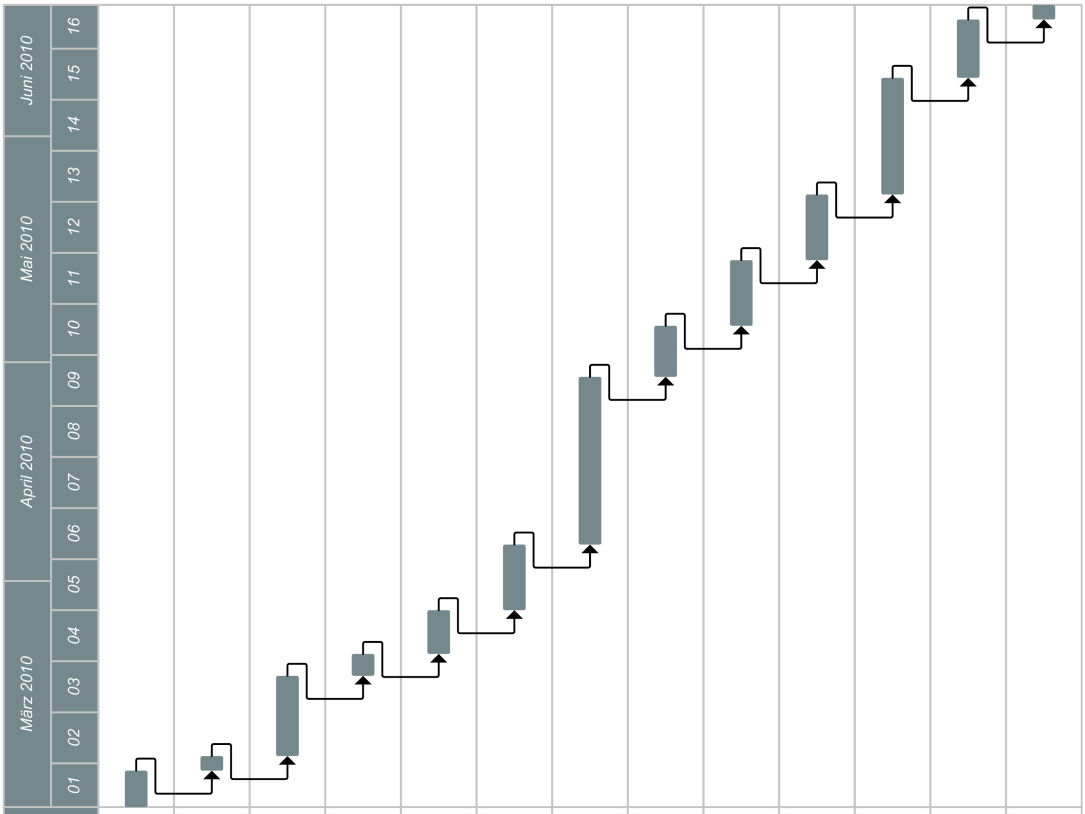


Illustration 1: Zeitplan

3 Meilensteine

Termin	*Meilensteine*	*Beschreibung*
01.03.2010	MS-1: Projektbeginn	Beginn des Projekts
08.03.2010	MS-2: Projektplanung & Zeitplanung	Projektplanung und Zeitplanung erstellt
29.03.2010	MS-3: Anforderungen	Anforderungen erfasst und Use Cases erstellt
05.04.2010	MS-4: Analyse & Design	Analyse und Design durchgeführt und Dokumentiert
12.04.2010	MS-5: Prototyp	Erster lauffähiger Prototyp entwickelt
17.05.2010	MS-6: Demonstrator	Demonstrator fertiggestellt
24.05.2010	MS-7: Dokumentation Arbeit	Dokumentation der Arbeit fertiggestellt
14.06.2010	MS-8: Dokumentation Lehre	Dokumentation für die Lehre fertiggestellt
18.06.2010	MS-9: Abgabe	Ende des Projekts

Table 1: Meilensteine

4 Soll-Ist Zeitvergleich

4.1.1 Zeitaufwände pro Woche

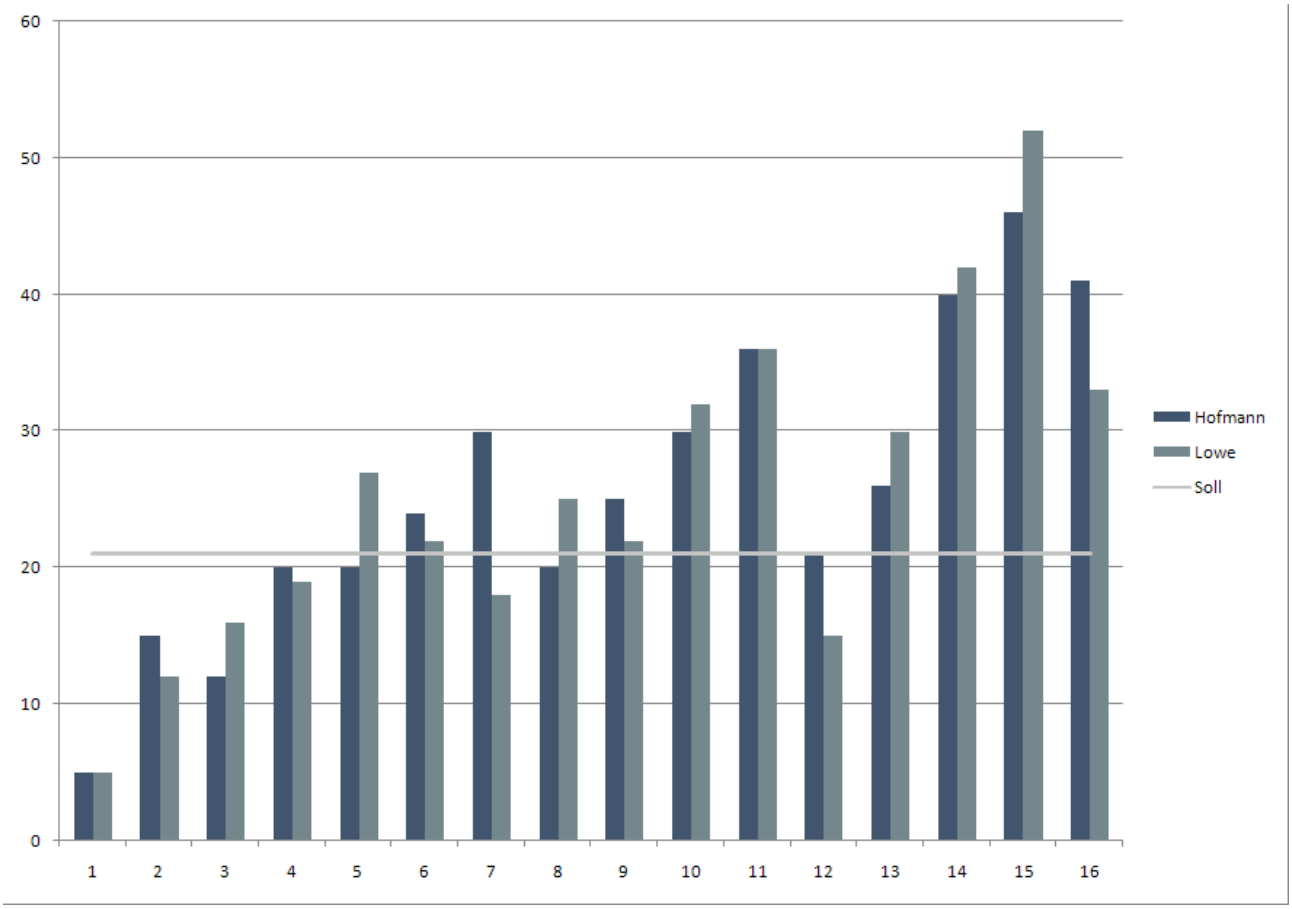


Illustration 2: Zeitaufwände pro Woche und Person

4.1.2 Zeitaufwände pro Arbeitspaket

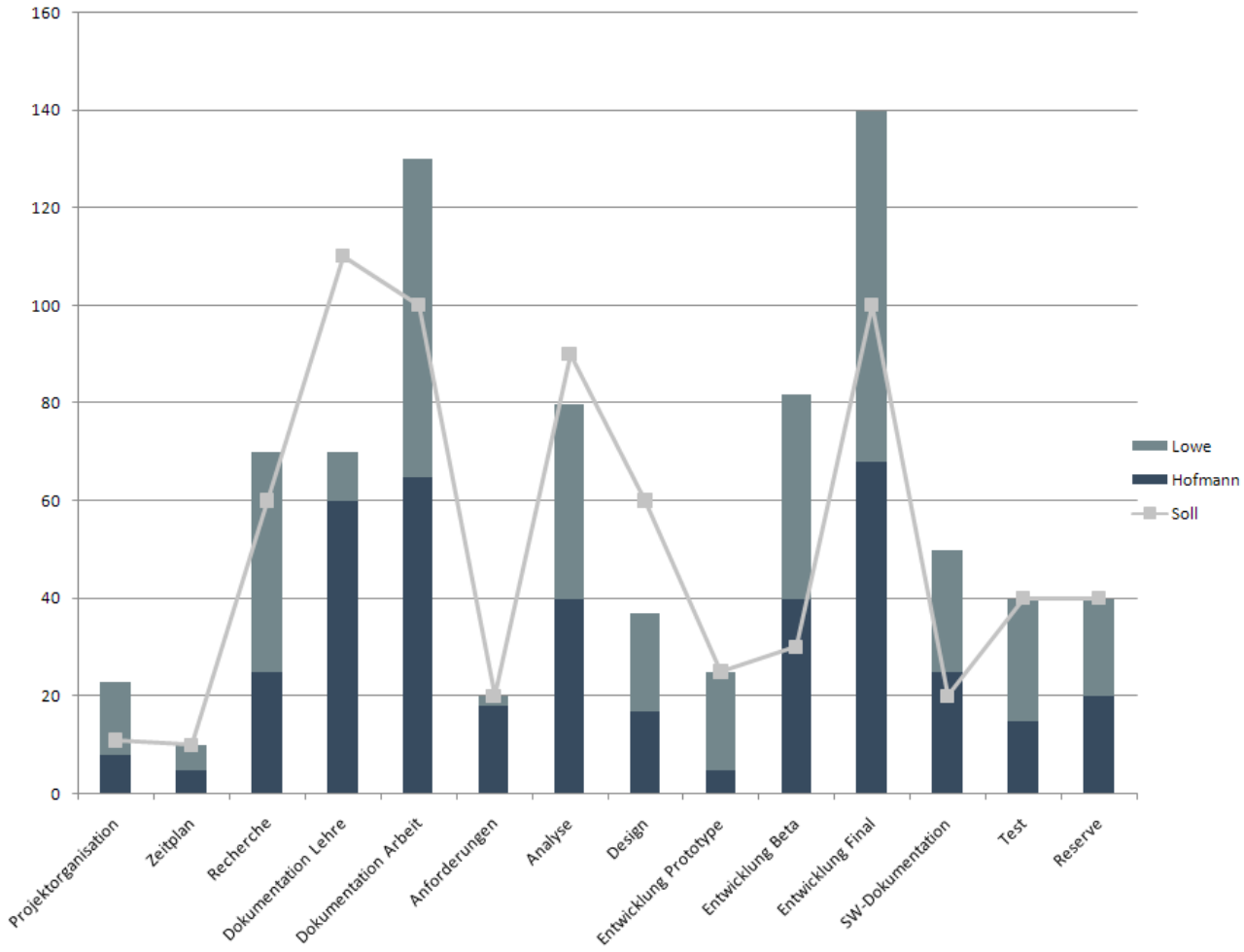


Illustration 3: Zeitaufwände Pro Arbeitspaket und Person

5 Anhang

5.1 Abbildungsverzeichnis

5.2 Tabellenverzeichnis

5.3 Literaturverzeichnis

Elektronische Versichertenkarte

Glossar

Status	Freigegeben
Klassifikation	Intern
Erstellt	2010-03-28
Geändert	2010-06-18
Besitzer	Hofmann Michael, Lowe Mark

Inhaltsverzeichnis

1 Einleitung

1.1 Inhalt

Glossar des gesamten Projekts.

1.2 Zweck

Liste von Begriffen mit Beschreibung.

1.3 Gültigkeit

Dieses Dokument besitzt keine Beschränkung der Gültigkeit.

2 Glossar

Begriff	*Beschreibung*
Administrative Daten	Daten wie Name, AHV-Nr., Versicherungsgesellschaft,...
Medizinische Daten	Daten wie Blutgruppe, letzte Impfungen, Allergien,...
VVK (-EDI)	Verordnung über die Versichertenkarte (Spezifikation zu Aussehen und Technischen Anforderungen)
eCH	Verein, welcher Standards zu Schweizerischem eGovernment herausgibt.
KVG	Krankenversicherungsgesetz
VK	Versichertenkarte
VeKa	Versichertenkarte
EDI	Eidgenössisches Departement des Innern
BAG	Bundesamt für Gesundheit
HPC	Health Professional Card (Karte zur Authentifizierung von Leistungserbringern)
FMH	Schweizerische Ärztevereinigung
VeKa	Versichertenkarte
PDC	Patient Data Card (Versichertenkarte)
APDU	Application Protocol Data Unit: Befehl, der vom Kartenlesegerät / Computer an das SmartCard- Betriebssytem gesendet wird.
JRE	Java Runtime Environment
HIC	Health Insurance Card (Versichertenkarte)

Table 1: Glossar

3 Anhang

3.1 Abbildungsverzeichnis

3.2 Tabellenverzeichnis

3.3 Literaturverzeichnis