Disease Management Programme auf Basis der Clinical Document Architecture (CDA)

Guido Noelle, Markus Hettlage, Stefan Topka
FH Bonn-Rhein-Sieg, FB Angewandte Informatik, St. Augustin

Einleitung

Die Einführung von Disease Management Programmen (DMP) für ausgewählte chronische Krankheiten ist seit geraumer Zeit in der gesundheitspolitischen Diskussion. Nachdem nunmehr mit Anlaufschwierigkeiten und zeitlichen Verzögerungen die Inhalte für Krankheitsbilder wie das Mamakarzinom und Diabetes mellitus Typ II verabschiedet wurden, befinden sich DMP-Festlegungen für weitere Krankheitsbilder wie die koronare Herzkrankheit oder Asthma/COPD noch in der Diskussion. Es ist dabei anzunehmen, dass die Palette der DMPs kontinuierlich auf ca. 10-15 verschiedene Krankheitsbilder ausgebaut wird.

Unter Disease Management ist ein integrativer Ansatz zu verstehen, der die episodenbezogene, sektoral aufgesplitterte Versorgung von einzelnen chronisch Kranken durch eine evidenzbasierte, Sektoren übergreifende und kontinuierliche Versorgung eines Patientenkollektivs von chronisch Kranken über alle Krankheitsstadien und Versorgungseinrichtungen hinweg ersetzt (Lauterbach et al., 2001). Im engeren Sinne dient die derzeitige Einführung der DMPs jedoch einer gerechteren Gestaltung des Risiko-Strukturausgleiches (RSA) der Krankenkassen. Der damit verbundene Dokumentationsaufwand ist ebenso erheblich wie die Weiterverarbeitung der dabei anfallenden Datenmengen.

Clinical Document Architecture und Disease Management

Die Kassenärztliche Bundesvereinigung (KBV) hat in 2003 damit begonnen, auf Grundlage der bestehenden Dokumentationsbögen Datensatzformate zu entwickeln, die künftig auch eine elektronische Erfassung und Weiterverarbeitung möglich machen sollen. Dabei wurde erstmals auf die Verwendung des im ambulanten Bereich sonst verbreiteten xDT-Datensatzformates verzichtet und an Stelle dessen konsequent auf XML gesetzt. Grundlage der vorliegenden Spezifikationen (siehe auch unter www.kbv.de) ist vielmehr die Verwendung der Clinical Document Architecture sowie der im Rahmen der Arbeitsgemeinschaft Sciphox erarbeiteten deutschen Anpassungen (siehe unter www.sciphox.de). Für die Softwarehäuser bedeutet dies einen erheblichen Paradigmenwechsel. Während sich zwar vielerorts in den vergangenen Jahren Initiativen und Projekte entwickelt haben, die XML im Gesundheitswesen einsetzen (an erster Stelle zu nennen ist hier die D2D Initiative der KV Nordrhein), sind die DMPs die ersten „amtlichen" Geschäftsvorfälle, die sich der CDA bedienen. Es ist zu vermuten, dass die KBV damit einen allmählichen Formatwechsel einleitet, der in wenigen Jahren dazu führen wird, den xDT-Standard vollständig zu Gunsten von XML ab zu lösen. Die Vorteile dabei liegen auf der Hand. XML ist ein Format, das flexibler und erweiterbarer ist, Binärdaten enthalten kann und kommunikationstauglicher ist. Überdies hinaus lassen sich Definitionen z.B. mit XML Schema verbindlicher abbilden und prüfen (siehe hierzu auch den Beitrag „Plausibilisierung von CDA-Dokumenten mit Hilfe von XSLT als Instrument der Qualitätssicherung"). Da CDA als ein Teilprodukt künftiger HL7-Entwicklungen anzusehen ist, ist neben dem wichtigen Aspekt der Internationalität darüber hinaus auch mittelfristig eine verbesserte Zusammenarbeit zwischen ambulanter und stationärer EDV zu erwarten.

Anwendungen, die heute bereits CDA einsetzen, werden in der Regel verschiedene Anzeigefenster programmieren, die eine Ansicht und Bearbeitung der spezifischen Formularinhalte möglich macht und diese in einer Datenbank ablegt. Die Erstellung eines CDA-Dokumentes aus den gespeicherten Daten wie auch der Import eines (fremden) CDA-Dokumentes in die Datenbank werden über spezifische Routinen gehandelt (Abbildung 1).

Dieses Vorgehen ist relativ aufwendig. So dürften die Inhalte der DMP-Bögen in aller Regel so speziell sein, dass die Daten bis auf wenige Ausnahmen nicht automati-

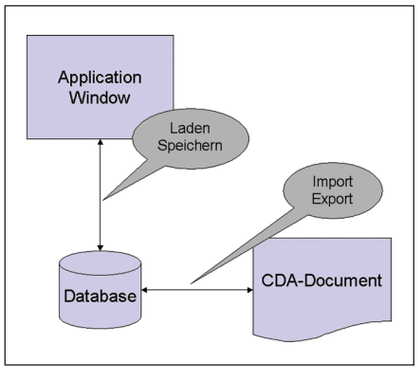

Abbildung 1: *Erstellung, Export und Import von CDA-Dokumenten*

Autoren: Guido Noelle, Markus Hettlage, Stefan Topka
Titel: Disease Management Programme auf Basis der Clinical Document Architecture (CDA)
In: Jäckel (Hrsg.) Telemedizinführer Deutschland, Ober-Mörlen, Ausgabe 2004
Seite: 128-131

Kap_3_3_TMF_04 neu 128 06.02.2006, 9:06 Uhr

siert aus anderen Teilen der elektronischen Dokumentation übernommen werden können. Weiterhin ist der Anpassungsbedarf bei – zu erwartenden – Änderungen als sehr hoch anzusehen; dabei müssen ggf. jeweils Datenbankstruktur, Anzeigemasken, Speicher-, Import- und Exportroutinen angepasst werden. Insbesondere bei der langfristig zu erwartenden Zahl verschiedener DMPs und weiterer Anwendungen, die auf CDA aufbauen, sind hier flexible und anpassungsfähige Lösungen gefragt, die den Pflegeaufwand der Anwendungen wirtschaftlich vertretbar machen.

Der Wunsch einer möglichst generischen Lösung für die automatisierte Erzeugung von Bearbeitungsfenstern aus zum Beispiel einer XML-Schema-Definition heraus besteht sicher bei vielen Entwicklern, konnte aber bislang nicht zufriedenstellend verwirklicht werden. Es gibt zwar mittlerweile eine Reihe von Werkzeugen, die halbautomatische Erstellung von Bildschirmmasken auf XML-Daten unterstützen. Dennoch sind diese Entwicklungen erst am Anfang und nicht generisch genug. Obgleich mit Hilfe des Schema-Objekt-Modells (SOM) oder der Verwendung der Extensible Stylesheet Language (XSL/T) mittlerweile gute Zugriffs- und Analysemöglichkeiten einer Schemadefinition bestehen, scheitert die automatisierte und generische Erstellung aus einer XML-Schema Definition heraus in aller Regel schon daran, dass eine Dokumenteninstanz immer endlich ist (und sein muss), Schemata jedoch durchaus Definitionen enthalten können, die sich gegenseitig aufrufen und somit bei einer automatisierten Instanzerzeugung zwangsläufig zu Endlosschleifen im Sourcecode führen (Eine Tabelle enthält etwa Zeilen mit Zellen, die wiederum eine Tabelle enthalten kann, die Zeilen mit Zellen, enthält, in denen Tabellen enthalten sind, etc.). Hinzu kommen noch die typischen Layoutfragen der Anzeige- und Eingabeelemente wie Positionierung, Größe, Farbe und Schrift. Ebenso unterliegen gerade die Datensatzbeschreibungen der DMPs einer Vielzahl von Feldabhängigkeiten, Regeln und Plausibilisierungen. Solche Prüfungen sollen in der Regel ja möglichst zeitnah bereits im Rahmen der Datenerfassung erfolgen.

Generische Implementierung mit XSLT

Im Rahmen einer generischen Prototypentwicklung für den Umgang mit CDA-Dokumenten wurden daher besonders die angesprochenen Probleme behandelt. Ein „hart kodiertes“ Framework, das unter Microsoft Windows Betriebssystemen als ausführbares Programm oder als COM-Komponente zur Einbindung in andere Anwendungen zur Verfügung steht, können beliebige CDA-Anwendungen dynamisch eingebunden werden. Die Rahmenanwendung hostet den Microsoft Internet Explorer für das GUI. Für die Verarbeitung der XML-Daten wird der Microsoft XML-Parser verwendet.

Bei der Verwendung als ausführbares Programm besteht die Möglichkeit, GDT- und BDT-Daten aus anderen Programmen zu verarbeiten. Dabei werden die Inhalte dieser xDT-Dateien dynamisch einzelnen Feldern zugewiesen. Die Inhalte umfassen dabei insbesondere die Versichertenstammdaten. Beim Speichern eines Formulares wird eine Referenz auf das gespeicherte Dokument im xDT-Format zurück gegeben. Eine Praxis-EDV Anwendung kann somit über in der Regel vorhandene Schnittstellenprotokolle einfach angebunden werden. Über die Referenz auf ein Dokument kann die Praxis-EDV das Dokument entweder nativ eigenständig weiterverarbeiten, im Browser aufrufen oder mit der CDA-Anwendung wieder aufrufen und bearbeiten. Da das Programm auch als Hintergrundanwendung in der Taskleiste lauffähig ist, muss es dabei nicht bei jedem Aufruf oder neuer Übergabe von Patientendaten im xDT-Format neu gestartet werden.

Die Erzeugung der Bildschirmanzeige erfolgt mit Hilfe von XSLT. Für jede spezifische CDA-Anwendung existiert ein separates Stylesheet, das sich aber über entsprechende „include“ und „import“ Anweisungen aus mehreren zusammensetzt. Damit ist es möglich, überschneidende bzw. immer gleiche Bereiche wie den Kopfteil der Formulare nur einmal zu erstellen, aber mehrfach zu verwenden. Jedes Dokument kann im Bearbeitungs- oder Anschaumodus angezeigt werden. Dazu wird das Stylesheet vor der Transformation entsprechend parametrisiert.

Das folgende Beispiel zeigt die Erzeugung eines Eingabe- oder Anzeigefeldes für die Patientennummer mittels XSLT. Dabei wird ein Template aufgerufen und mit bestimmten Vorgaben für die Anzeige und Formatierung belegt. Im Parameter „value“ wird der Wert aus dem CDA-Dokument abgefragt und übergeben. Wenn es sich um ein neues Dokument handelt, wird ein Standard-CDA-Dokument für den Anwendungsfall transformiert. Damit lassen sich beispielsweise sehr einfach Standardbelegungen für Felder erzeugen. Über das zusätzliche Attribut „bdtRef“ kann eine automatische Verbindung zu übergebenen Feldinhalten einer xDT Datei hergestellt werden. So stellt die Feldkennung „3000“ in einer xDT Datei immer die Patientennummer dar. Beim Einlesen der xDT-Datei werden die Felder zeilenweise abgearbeitet und geschaut, ob ein Eingabefeld mit einem zur Feldkennung referenzierenden „bdtRef“-Attribut besteht und der „value“ des Feldern dann mit dem Inhalt aus der xDT Datei gefüllt. (Tabelle 1)

```
<xsl:call-template name="createTextInput">
    <xsl:with-param name="id">patientennummer</xsl:with-param>
    <xsl:with-param name="style">width:116px;top:130px;left:441px</xsl:with-param>
    <xsl:with-param name="bdtRef">3000</xsl:with-param>
    <xsl:with-param name="maxlength">6</xsl:with-param>
    <xsl:with-param name="value">
        <xsl:value-of select="//cda:patient//cda:id/@EX" />
    </xsl:with-param>
</xsl:call-template>
```

Tabelle 1

Kap_3_3_TMF_04 neu 129 06.02.2006, 9:06 Uhr

```
<xsl:if test="$typeOfProcessing = ‚edit'">
        <xsl:element name="input">
            <xsl:attribute name="value"><xsl:value-of select="$value" /> </xsl:attribute>
        </xsl:element>
    </xsl:if>
<xsl:if test="$typeOfProcessing = 'view'">
    <xsl:element name="div">
        <xsl:value-of select="$value" />
    </xsl:element>
</xsl:if>
```

Tabelle 2

```
<xsl:element name="id">
    <xsl:attribute name="EX">
        <xsl:value-of select=»//INPUT[@id='patientennummer']/@value» />
    </xsl:attribute>
    <xsl:attribute name="RT">
        <xsl:value-of select="//INPUT[@id='arztnummer']/@value" />
    </xsl:attribute>
</xsl:element>
```

Tabelle 3

hinterlegen und beliebig wieder verwenden, ohne dass diese Felder jeweils von Hand ausgefüllt werden müssen.

Während sich die Diskussion über die Verarbeitung von XML-Dokumenten häufig auf das Problem der Generierung geeigneter Eingabemasken beschränkt, stellt das Speichern ebenso ein Problem dar, das zumeist nur durch „harte" Codierung und dem damit verbundenen Anpassungsaufwand verbunden ist. In dem vorgestellten Ansatz wurde ein anderer Weg gewählt. Vor dem Speichern wird das Anzeigeformular in xHTML umgewandelt. Damit kann es dann wiederum mit Hilfe eines Stylesheets in ein gültiges CDA-Dokument transformiert werden. Der folgende Ausschnitt zeigt die Erzeugung des „<id>"-Elementes, das in CDA die Patientennummer beschreibt. Dazu werden aus dem xHTML Dokument die Werte aus den entsprechenden Eingabefeldern per xPath extrahiert und in das entstehende CDA-Dokument eingefügt (Tabelle 3).

3

Das aufgerufene Template (Ausschnitt) unterscheidet nun, ob das Dokument im Bearbeitungsmodus angezeigt werden soll oder nicht. Ensprechend wird ein „<input>" Feld oder ein „ " Tag erzeugt (Tabelle 2).

Der aufwendigste Teil bei der Gestaltung eines solchen Stylesheets ist die genaue Festlegung der Positionen der Anzeigeelemente. Hierfür wurde ein spezieller Editor entwickelt, der die Erzeugung und . – genaue – Positionierung im WYSIWYG-Modus im Browserfenster per Mausklick und Drag&Drop ermöglicht und das Resultat als Stylesheet abspeichert. Darüber hinaus können innerhalb des Editors u.a. auch die notwendigen Verknüpfungen zu einem CDA-Dokument festgelegt werden. Auch aufwendige Bildschirmmasken lassen sich so in kurzer Zeit effektiv erstellen. Über den Verweis „document(URL)" lassen sich innerhalb von Stylesheets während der Transformation auch weitere XML-Dateien öffnen. So lassen sich Daten wie der „Arztstempel" einmalig in einer XML-Parameterdatei

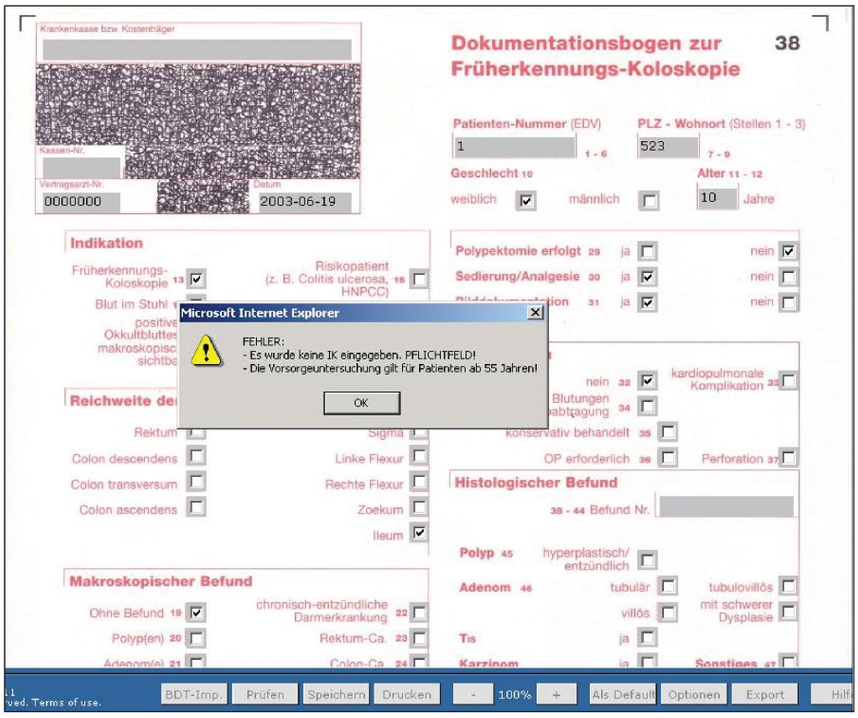

Abbildung 2: *Plausibilitätsprüfungen mit XSLT im Rahmen des Speichervorganges am Beispiel der Früherkennungs-Koloskopie*

Kap_3_3_TMF_04 neu 130 06.02.2006, 9:06 Uhr

Quartal	Datum	Bericht	Patient	Kasse (IK)	Arzt	Check	Export	Delete	Dateiname	Nr.	Edit
2003-2	2003-06-26	Koloskopie-Bericht			0000000	Fehler		X	rad7599C.M38	10	X
2003-2	2003-06-26	Koloskopie-Bericht			0000000	Fehler		X	radDB9F3.M38	11	X
2003-2	2003-06-19	Koloskopie-Bericht	1		0000000	Fehler		X	rad1530E.M38	9	X
2003-2	2003-06-06	Koloskopie-Bericht	1		0000000	Fehler		X	rad22C94.M38	8	X
2003-2	2003-06-05	Koloskopie-Bericht			0000000	Fehler		X	rad6EA66.M38	7	X
2003-2	2003-05-28	Koloskopie-Bericht			0000000	Fehler		X	rad5D563.M38	6	X
2003-2	2003-05-15	Koloskopie-Bericht	234567	4536677	0000000	OK		X	rad46451.M38	5	X
2003-2	2003-05-12	Koloskopie-Bericht			0000000	Fehler		X	radB1B7C.M38	1	X
2003-2	2003-05-12	Koloskopie-Bericht			0000000	Fehler		X	rad88972.M38	3	X
2003-2	2003-05-12	Koloskopie-Bericht			0000000	Fehler		X	rad00CF8.M38	4	X
2002-4	2002-12-12	Koloskopie-Bericht			0000000	Fehler		X	rad0F6FE.M38	2	X

Abbildung 3: *Protokollübersicht der erstellten CDA-Dokumente*

Innerhalb dieses Stylesheets sind dann zusätzlich die erforderlichen Plausibilitätsprüfungen und von der KBV vorgeschriebenen Regeln implementiert (siehe hierzu auch den Beitrag „Plausibilisierung von CDA-Dokumenten mit Hilfe von XSLT als Instrument der Qualitätssicherung“). Das CDA-Dokument kann wahlweise in einer Datenbank oder als File gespeichert werden, nachdem es zusätzlich erfolgreich gegen ein XML-Schema validiert wurde. Damit wird sowohl die formale als auch inhaltliche „Richtigkeit“ des Dokumentes vor jedem Speichervorgang überprüft (Abbildung 2).

Alle Fehlermeldungen werden in einer Protokolldatei vermerkt. Fehlerhafte Dokumente können zwar gespeichert werden, aber nicht etwa im Rahmen der quartalsweisen Übermittlung an die KV übermittelt werden (Abbildung 3).

Zusammenfassung

Der vorgestellte Ansatz zur Erzeugung und Bearbeitung unterschiedlicher CDA-Anwendungen zum Beispiel im Bereich von DMP ist generisch und spart Entwicklungs- und Pflegeaufwand. Es wurde eine Containeranwendung mit übergreifenden Funktionen für die Verwaltung von CDA-Dokumenten (Anzeigen, Speichern, Bearbeiten, Drucken, ...) und Standard-xDT-Schnittstellen zur Kommunikation mit beliebigen Praxis-EDV-Anwendungen entwickelt. In dieser können mit Hilfe von XSLT beliebige CDA-Geschäftsvorfälle vollständig eingebunden werden, ohne die Rahmenanwendung hierfür jeweils anpassen zu müssen. Damit ist die Anwendung leicht erweiterbar und pflegbar. Die erzeugten Dokumente können neben einer Validierung gegen XML-Schema zusätzlich inhaltlich gemäß evt. vereinbarter Geschäftsregeln plausibilisiert werden.

Kontakt

Dr. med. Guido Noelle
Gesundheitsökonom, Professurvertreter
für Medizinische Informatik
Fachhochschule Bonn-Rhein-Sieg
University of Applied Sciences
Fachbereich Angewandte Informatik
Department of Applied Computer Science
Grantham-Allee 20
D-53757 Sankt Augustin, Germany
Tel.: +49(0)2241-865-239
Fax: +49(0)2241-865-8217
Email: guido.noelle@fh-bonn-rhein-sieg.de
www.inf.fh-rhein-sieg.de/person/professoren/noelle/index.htm

Kap_3_3_TMF_04 neu 131 06.02.2006, 9:06 Uhr