Erfahrungsbericht über den Einsatz des open source Kommunikationsservers Mirth Connect am Universitätsklinikum Bonn

P. Köppen, S. Langenberg
uk-it Köln Bonn
Universitätsklinikum Bonn
Sigmund-Freud-Str. 25
53105 Bonn
Philipp.Koeppen@ukb.uni-bonn.de
Stefan.Langenberg@ukb.uni-bonn.de

Abstract: Kommunikationsserver spielen im Krankenhaus zur Verknüpfung heterogener medizinischer Anwendungssysteme eine zentrale Rolle. Seit 2013 wird am Universitätsklinikum Bonn der open source Kommunikationsserver Mirth Connect eingeführt. Mit zusätzlichen, nicht einer open source Lizenz unterliegenden Erweiterungen ist das System für den Produktiveinsatz im Krankenhaus geeignet.

1 Einleitung

Derzeit gibt es kein Informationssystem das alle Funktionen, die in einem Krankenhaus benötigt werden, bereitstellen kann. [KG01] In vielen Einrichtungen existieren neben einem eher administrativ ausgerichteten Patientenmanagementsystem und klinischen Dokumentationssystem spezialisierte Subsysteme, z.B. für das Labor oder die Radiologie. Diese müssen über Schnittstellen miteinander kommunizieren, um effektive anwendungsübergreifende Arbeitsprozesse zu ermöglichen. Am UKB wurde dazu ein Prozess orientiertes Integrationskonzept in einer Nachrichten basierten Architektur gewählt [CHKT06]. Hierzu hat sich der Kommunikationsstandard HL7 im Krankenhaus etabliert [HBD99].

Mittels HL7 lassen sich zwar grundsätzlich Kommunikationsbeziehungen zwischen den einzelnen Anwendungen direkt realisieren, wie dies beispielsweise am Universitätsklinikum Freiburg im Sinne einer HL7-basierten serviceorientierten Architektur realisiert wurde. Um jedoch die Komplexität der Kommunikationsbeziehungen zu reduzieren, kommen Kommunikationsserver zum Einsatz [Bra10]. Sie haben folgende Funktion: „Ein Kommunikationsserver überwacht alle Verbindungen zwischen den Applikationen in einer Organisation. Eine Verbindung wird zwischen jedem System und dem Kommunikationsserver hergestellt und die Nachrichten zwischen diesen bidirektionalen Verbindungen können übersetzt, gefiltert, archiviert, modifiziert und zu ihrem endgültigen Ziel weitergeleitet werden.“ [Spr07]. Treffender ist daher die englische Bezeichnung *integration engine*.

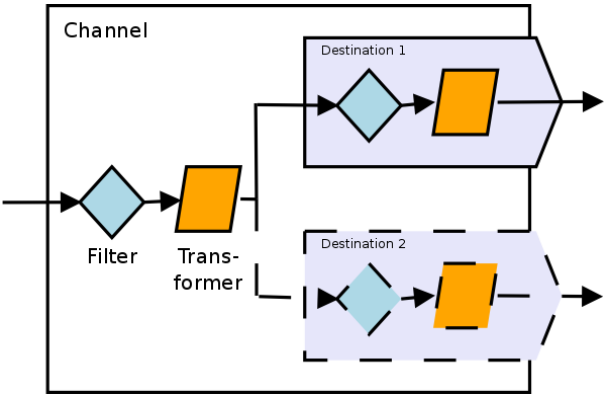

Abbildung 1: Channel mit Quellkonnektor, allen möglichen Filtern und Transformatoren sowie zwei Zielkonnektoren

Der Kommunikationsserver Mirth Connect wurde von der Mirth Corporation, die von Quality Systems, Inc. im Jahre 2013 übernommen wurde entwickelt [QSI]. Der Java-Quellcode wurde unter der Mozilla Public License (MPL) 1.1 veröffentlicht.

Kommunikationsbeziehungen werden mit Mirth Connect mittels sog. Channel verwirklicht [HW03]. Mindestens einen Channel benötigt man, um Daten von A nach B zu transportieren. Jeder Channel läuft als Thread innerhalb eines Mirth Connect Prozesses. Jeder Channel hat einen Eingang (Quellkonnektor), über den Daten von einem externen System entgegengenommen werden können. Weiterhin kann jeder Channel auch Daten von einem oder mehrerer anderer Channel entgegennehmen, wodurch sich auch komplexere Kommunikationsbeziehungen umsetzen lassen. Für die verarbeiteten Daten gibt es einen oder mehrere Zielkonnektoren (destinations).

Jeweils auf der Eingangs- wie Ausgangsseite befinden sich Filter und Transformatoren zur internen Verarbeitung der Nachrichten, siehe Abb. 1. Intern werden die Nachrichten nach XML konvertiert. Empfangene und gesendete Nachrichten werden für einen einstellbaren Zeitraum in einer Datenbank gespeichert, wobei die Datenbanken Apache Derby, MySQL, PostgreSQL und MS SQL unterstützt werden. Mittels des in Java geschriebenen JavaScript Interpreters Rhino [Rhi] sind eigene Programmentwicklungen innerhalb von Mirth Connect ausführbar. Die Administration des Systems erfolgt über eine per Java Web Start aufrufbare graphische Benutzerschnittstelle, siehe Abb 2.

2 Ergebnisse und Erfahrungen

Seit Mitte 2013 verfügt das Universitätsklinikum Bonn über eine Produktions- und Entwicklungsumgebung einer Mirth Connect 2.0.5 Installation mit von der Firma OSM-GmbH entwickelten Erweiterungen (Mirth Connect powered by OSM), diese Erweite-

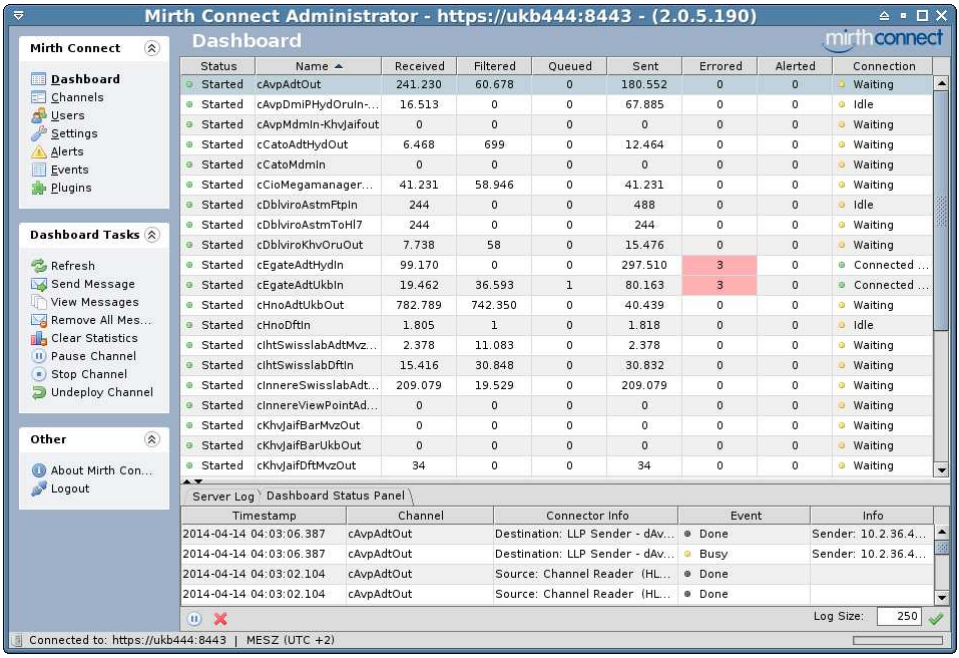

Abbildung 2: Mirth Connect Dashboard mit Ansicht aller Channel

rungen unterliegenden jedoch nicht einer open source Lizenz. Darüber hinaus verwenden mindestens 4 klinische Subsysteme Mirth Connect als Schnittstelle zu den zentralen Kommunikationsservern. Bisher wurde der Kommunikationsserver eGate 5-SRE verwendet. Da die Weiterentwicklung dieses Produktes seitens der Firma Oracle allerdings ungewiss ist, haben wir und auf Anraten der OSM-GmbH für den Umstieg auf Mirth Connect entschieden. Da der Support für eGate 5-SRE noch bis 2017 gesichert ist, besteht für den Umstieg kein Zeitdruck. Zunächst implementieren wir alle neue Schnittstelle unter Mirth Connect, nebenher stellen wir sukzessive die bestehenden Schnittstellen auf Mirth Connect um.

Die zentralen Mirth Connect Installationen werden unter Windows 2008 64bit mit MySQL als Datenbank betrieben.

Ein systematischer Vergleich von Mirth Connect mit dem bisher verwendeten eGate 5-SRE sowie dem Cloverleaf Kommunikationsserver ist in Tab. 1 dargestellt.

2.1 Protokollunterstützung

Mirth Connect verwendet XML als internes Datenformat zur Nachrichtenspeicherung. Andere Formate wie CSV, HL7, DICOM werden in den Quellkonnektoren nach XML gewan-

Tabelle 1: Vergleich von Mirth Connect mit zwei kommerziellen Kommunikationsservern mit hohem Marktanteil im Krankenhaus

	Mirth Connect	**eGate 5-SRE**	**Cloverleaf 5.8**
Konnektoren	Dateisystem, CIFS, (S)FTP, TCP/IP-Socket, HTTP, Mail, Web-Service (SOAP), JMS, DICOM, Datenbank, selbstentwickelte JavaScript Konnektoren	Dateisystem, FTP, TCP/IP-Socket, weitere als Zusatzmodul (z.B. SAP-BAPI, SAP-HCM)	Dateisystem, FTP, SFTP als Zusatzmodul, TCP/IP-Socket, HTTP, Mail, DICOM, selbstentwickelte Java oder Tcl Konnektoren
Nachrichtenprotokolle	XML, HL7 2.x, HL7 3.x, CSV, UN/EDIFACT, SAP-HCM (OSM Erweiterung)	XML, HL7 2.x, CSV, UN/EDIFACT, Frei definierbare Formate, SAP-HCM (OSM Erweiterung)	XML, HL7 2.x, HL7 3.x, HRL (Zusammenstellung aus VRL/FRL) , VRL (Variable Formate), FRL (Fixe Formate)
Nachrichtentransformation	JavaScript, XSLT	Monk (Scheme), XSLT, Java	Tcl, Java
Transaktionssicherheit	bedingt vorhanden	vorhanden	vorhanden
Alerting	z.T. vorhanden, sonst über internes JavaScript	über externe shell Skripte	Networkmonitor, Alertkonfigurator
Nachrichtenarchivierung	Datenbank	Dateisystem	Dateisystem
Dokumentation	Online Zugang nur bei Supportvertrag mit QSI	vollständig	vollständig

delt. Die XML-Strukturen werden für jede Nachricht einzeln dynamisch generiert, so dass auch nicht Standard konforme Nachrichten verarbeitet werden können.

Das für das Patientenmanagement verwendete SAP Modul IS-H verwendet den SAP eigenen Nachrichtenstandard HCM [SAP99]. Mit einer Programmerweiterung von OSM kann auch dieses verarbeitet werden.

Für von Mirth Connect nicht primär unterstützte Nachrichtenformate muss man sich mittels JavaScript einen Parser schreiben, um das Format nach XML zu konvertieren. So konnte mit Mirth Connect auch eine verbesserte Integration eines Virologie Laborsystems in einem ASTM-ähnlichen Befundformat realisiert werden.

Auf der Transportebene unterstützt Mirth Connect TCP/IP MLLP (meist verwendet im Zusammenhang mit HL7), dateibasierte Protokolle (File, SMB, FTP, SCP). Das Umbenennen von Dateien nach der Übermittlung wird von Mirth Connect erst ab Version 3 im Standard unterstützt. Mit diesem Mechanismus möchte man verhindern, dass eine Datei bereits vom Zielsystem verarbeitet wird, während sie noch geschrieben wird. Semaphore Dateien werden lediglich in der OSM-Erweiterung unterstützt.

Für den Import und Export werden weiterhin Web Services (SOAP, WSDL) sowie HTTP unterstützt.

Der direkte Zugriff auf Datenbanken ist ebenfalls möglich. So wurde die Übermittlung von Behandlungsdokumenten aus dem zentralen Dokumentenmanagementsystem (DMS) an ein Einweiserportal entwickelt. Mirth Connect fragt regelmäßig neue Eingänge von Dokumenten ab und übermittelt diese im HL7 Standard an ein Einweiserportal. Die Einbindung von Datenbanken in Kommunikationsprozesse ist mit Mirth sehr anwenderfreundlich möglich, da die notwendigen Datenbanktreiber bereits in die Software integriert sind. Eine SQL Abfrage, die in einem JavaScript eingebunden ist kann auch Abfrageparameter enthalten, die erst zur Laufzeit ermittelt werden.

```
// Get timestamp of last SQL execution
var strFile = "LastTimestamp.seq";
var contents = FileUtil.read(strFile);
var strLetzterAbfrageZeitpunkt = contents; // z.B.: "01032014100000";
// Get Data from Oracle SQL Database
var dbConn = DatabaseConnectionFactory.createDatabaseConnection( ... );
//Create a SQL Statement to be executed
var sqlStr =
"select ... FROM ... " +
"where ..." +
"BPD.DOKUMENTENDATUM >= TO_TIMESTAMP ('" + strLetzterAbfrageZeitpunkt
 + "', 'DDMMYYYYHH24MISS')";
// Execute SQL Statement and save them as results
var result = dbConn.executeCachedQuery(sqlStr);
dbConn.close();
return result;
```

Neben der Echtzeitübermittlung von HL7 Nachrichten, wird Mirth Connect auch zur Extraktion und Bereitstellung von Datenpools genutzt, die für Anwender aus den administrativen Bereichen (z.B. Medizincontrolling) als Basis für Auswertungen verwendet werden. Insbesondere SQL Abfragen mit längeren Antwortzeiten, werden mit Mirth Connect zu einem festgelegten Zeitpunkt in der Nacht ausgeführt und die gewonnenen Daten per FTP oder SMB auf Abteilungslaufwerke abgelegt. Somit werden die Abfragezeiten außerhalb der Arbeitszeit verlegt.

2.2 Nachrichtenfilterung und Übersetzung

Einfache Filter und Transformationen lassen sich mit der graphischen Benutzeroberfläche erstellen. Besser und flexibler ist dies jedoch mit dem integrierten JavaScript möglich, Beispiel siehe Abb. 3.

Eine MDM-T01 Nachricht mit base64-kodiertem eingebettetem Dokumenteninhalt lässt sich beispielsweise aus übermittelnden Dokumenten (PDF, TIFF, JPEG) in einem Transformationsprozess mittels JavaScript erzeugen.

```
var uncpath = "\\server\share\" + file;
var contents = FileUtil.readBytes(uncpath); // (PDF-)Datei einlesen per UNC-Pfad:

// Dokument base64 kodieren.
var encData  = FileUtil.encode(contents);

//remove the newlines to produce valid Base64
var index = encData.indexOf("\r\n");
    while(index != -1){
        encData = encData.replace("\r\n","");
        index = encData.indexOf("\r\n");
    }

// Base64 kodiertes (PDF-)Dokument in OBX.5.1 Feld schreiben.
tmp['OBX']['OBX.5']['OBX.5.1'] = encData;
```

Hl7-Nachricht:

```
MSH|^~\&|ORBIS_ADT|MVZ|MEDOS|RAD|201403241134||ADT^A34|1408963|P|2.3|||AL|NE|D||DE
EVN|A34|201403051010|201403051010||UKB17525ADM||
PID|1|6643112|06643112||Test^MVZ^^^^||19901010|M||
MRG|06655915|
```

Interne XML-Darstellung der HL7-Nachricht (Ausschnitt):

```
<HL7Message>
    <MSH>
        ...
        <MSH.4>
            <MSH.4.1>MVZ</MSH.4.1>
        </MSH.4>
        <MSH.5>
            <MSH.5.1>MEDOS</MSH.5.1>
        </MSH.5>
        <MSH.6>
            <MSH.6.1>RAD</MSH.6.1>
        </MSH.6>
        <MSH.7>
            <MSH.7.1>201403241134</MSH.7.1>
        </MSH.7>
        <MSH.8/>
        <MSH.9>
            <MSH.9.1>ADT</MSH.9.1>
            <MSH.9.2>A34</MSH.9.2>
        </MSH.9>
...
```

Zugriff auf Nachrichtenelemente mit JavaScript am Beispiel eines Filters:

```
var valid_Events = new
RegExp("A01|A02|A03|A04|A05|A06|A07|A08|A11|A12|A13");
var valid_Ambulance = new RegExp("M1551|M1552");
var valid_Institute = new RegExp("M15IS");

var process = false;

if (msg['EVN']['EVN.1']['EVN.1.1'].toString() == "A34") process = true;
else if (valid_Events.test(msg['EVN']['EVN.1']['EVN.1.1'].toString()))
{
if (valid_Ambulance.test(msg['PV1']['PV1.3']['PV1.3.4'].toString())
||

valid_Institute.test(msg['PV1']['PV1.3']['PV1.3.5'].toString()))  process =
true
}

return process;
```

Abbildung 3: Beispiel einer einfachen HL7-Nachricht und deren interne XML-Darstellung, sowie den Zugriff auf Feldinhalte in einer JavaScript Filterregel

2.3 Transaktionssicherheit

Verbesserungen bei der Transaktionssicherheit gibt es mit der Version 3. Bei Version 2 war diese lediglich für TCP/IP-Socketkommunikation gegeben. Wenn auf ein Zielsystem mit einem dateibasierten Transportprotokoll wie beispielsweise FTP eine Nachricht übertragen wird und das Zielsystem nicht verfügbar ist, so läuft die Verarbeitung der Nachricht auf einen Fehler. Seit Version 3, sowie in der speziellen OSM-Version, die wir verwenden, wird die Nachricht in einer Queue gepuffert und nach Wiederverfügbarkeit des Zielsystems automatisch nachkommuniziert.

Weitere Probleme bei der Transaktionssicherheit treten bei der Übertragung sehr großer Dateien auf. Bei erschöpftem Java heap space wird die Quelldatei zwar gelöscht, die Zieldatei aufgrund des Fehlers aber nicht geschrieben.

2.4 Monitoring und Alarme

Alle Fehlermeldungen werden in einer Log-Datei protokolliert. Die Zuordnung zu einem bestimmten Channel ist dabei meist schwierig. Pro Channel werden die letzten 200 (einstellbar) Log-Einträge gespeichert und über die graphische Benutzeroberfläche angezeigt. Solange eine Nachricht noch nicht nach der Verarbeitung aus der Datenbank gelöscht wurde, lässt sich über die Benutzeroberfläche der Erfolg der Verarbeitung kontrollieren.

Bei Fehlerzuständen lassen sich Email-Alerts konfigurieren. Bei dem o.g. Beispiel des nicht verfügbaren Zielsystems sind allerdings solche Alerts wenig hilfreich, wird doch bei jedem Versuch einer fehlerhaften Nachrichtenübermittlung eine Email verschickt. An dieser Stellen kann man allerdings sich mit JavaScript behelfen: Innerhalb eines Channels kann per JavaScript die Anzahl der Nachrichten in der Queue ermittelt werden, die nicht zu einem Zielsystem kommuniziert werden konnten. Erst wenn diese Anzahl einen bestimmten Schwellwert überschreitet, wird eine Fehlermeldung versandt. Nach Behebung des Fehlers beim Zielsystem kann dann nach Unterschreitung des Schwellwerts per Email der Alarm wieder aufgehoben werden. Ein solcher Mechanismus ist allerdings nur anwendbar, wenn für jedes Zielsystem ein eigener Channel zuständig ist, siehe Abb 4.

2.5 Archivierung

Für einen gewissen Zeitraum lassen sich die bereits kommunizierten Nachrichten in der Datenbank von Mirth Connect aufbewahren, z.B. für 7 Tage. Für die Langzeitarchivierung ist dieses Verfahren aufgrund des benötigten Speicherbedarfs nicht geeignet. Man kann das Problem wie folgt lösen: Die Verarbeitung, die eigentlich über einen Channel abgewickelt werden könnte, wird in einen Quell- und Zielchannel aufgespalten. Wir senden daher aus jedem Quellchannel über einen Archivkonnektor alle Nachrichten eines Tages in eine Archivdatei. Bei Bedarf lassen sich einzelne Archivdateien oder Teile davon über einen Resend-Channel an den gewünschten Zielchannel nachkommunizieren, siehe Abb. 4. Ein

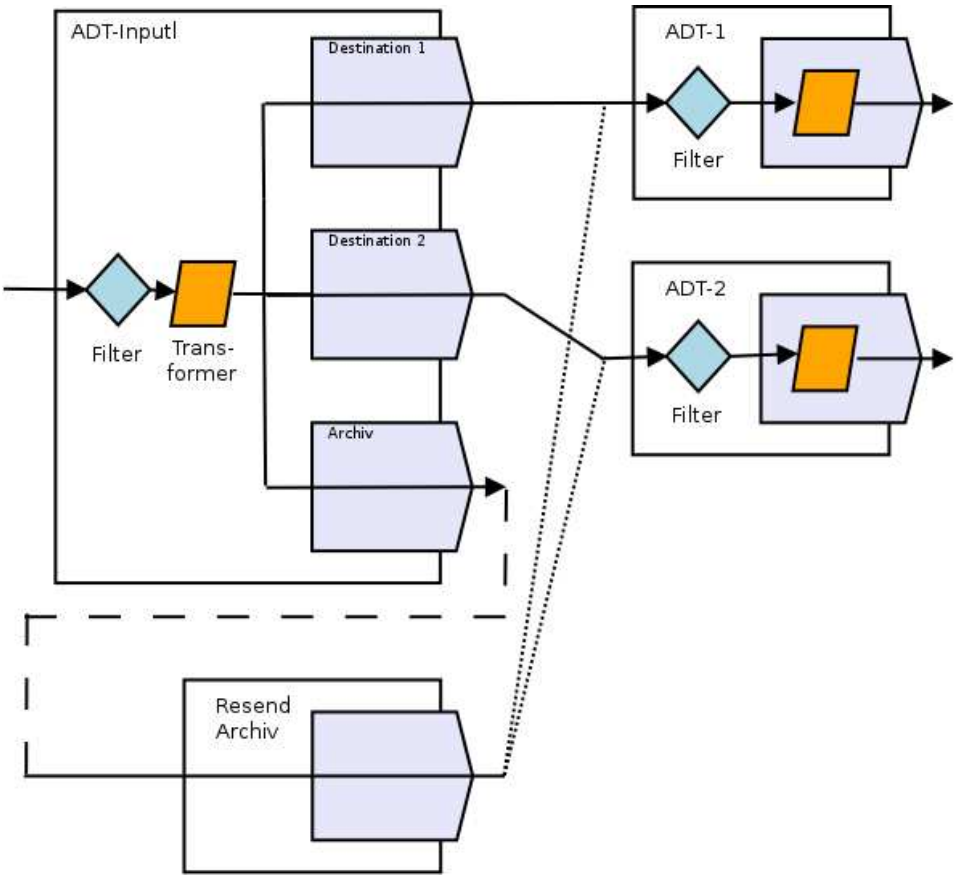

Abbildung 4: Nachrichtenverarbeitung mit mehreren Channeln am Beispiel des Transfers von Patientenstamm- und Bewegungsdaten von einem zentralen Patientenmanagementsystem an mehrere Subsysteme (ADT-1 und ADT-2). Im Quellchannel werden die Nachrichten zusätzlich über einen Archivkonnektor tageweise gebündelt als Archivdatei abgelegt. Bei Bedarf können diese über einen Resend-Channel an den gewünschten Zielchannel nachkommuniziert werden. Dabei durchlaufen sie in dem Zielsystemchannel die gleichen Filter- und Transformationsschritte wie die ursprüngliche Nachricht.

vergleichbarer Mechanismus für jeden Channel wird mit Version 3 im Standard bereitgestellt.

2.6 Dokumentation

Das grobe Fundament für den fachlichen Einstieg in die Mirth Connect Umgebung wurde durch die Basisschulung der OSM-GmbH gelegt. Wesentlich für den Aufbau von Wissen zu Schnittstellenentwicklung ist jedoch die sogenannte Community-Plattform des Herstellers in den USA. Es besteht die Möglichkeit sich kostenfrei an der Plattform anzumelden und somit Einsicht in zahlreiche Anfragen und Problemlösungen zu erhalten. Der Umfang der Forenbeiträge ist sehr umfangreich und zu den meisten Fragestellungen konnte eine Lösung gefunden werden, die ggf. leicht angepasst werden musste. Die Beträge sind durchweg in englischer Sprache verfasst. Je nach Problem gestaltet sich die Suche nach einem passenden Beitrag mit Lösungsansatz recht langwierig. Mirth Connect verwendet intern die Skriptsprache JavaScript, deren Programmierkonzepte und Syntax sehr gut dokumentiert ist. Daher konnte die Umsetzung von Anforderungen zu Formatanpassungen bei Schnittstellenprojekten in Programmcode schnell umgesetzt werden. Mirth Connect bietet die Möglichkeit die konfigurierte Schnittstellenlogik in Gesamtheit oder einzeln in einer XML Konfigurationsdatei abzuspeichern. Diese Datei kann von einem anderen Mirth Connect-System importiert werden und sofort ist die gleiche Schnittstelle implementiert. Dies erleichtert den Austausch mit anderen Nutzern erheblich: Realisierte Schnittstellenlösungen können als konkrete Implementierungsvorlage schnell und bequem bereitgestellt werden; offene Implementierungsprobleme werden durch Konfigurationsentwürfe konkretisiert. Dieser Nutzen ist jedoch stark abhängig von der Partizipationsbereitschaft und dem allgemeinen Wissenstand innerhalb der Nutzergemeinde.

3 Fazit

Mirth Connect v2 mit den von OSM bereitgestellten Erweiterungen kann unseren bisherigen Kommunikationsserver eGate 5 weitestgehend ersetzten. Mit Version 3 sind viele dieser Erweiterungen bereits im Standard enthalten.

Mit Mirth Connect lassen sich Schnittstellen mit deutlich geringerem Aufwand als mit eGate realisieren. Erste produktiv verwendete Schnittstellen konnten von eGate auf Mirth Connect umgestellt werden.

Durch den open source Charakter der Software konnten Lizenzkosten für die Beschaffung eines neuen Kommunikationsservers eingespart werden. Durch den kostenpflichtigen Support eines Drittanbieters ist das Betriebsrisiko nicht höher als bei der Verwendung eines kommerziellen Produktes. Die rasche Verbreitung, die Mirth Connect zwischenzeitlich international gefunden hat, lässt eine dauerhafte und dynamischen Weiterentwicklung erwarten.

Um den Austausch zwischen den Mirth Connect Benutzern im deutschsprachigen Raum

zu verbessern, haben wir eine deutschsprachige Internetplattform zu Mirth Connect ins Leben gerufen [Lan].

Literatur

[Bra10] E.M. Brauer. Haben Kommunikationsserver eine Zukunft? - Künftige Anforderungen und Kriterien. *Krankenhaus-IT Journal*, 4:53–54, 2010.

[CHKT06] S. Conrad, W. Hasselbring, A. Koschel und R. Tritsch. *Enterprise Application Integration*. Elsevier, 1. Auflage, 2006.

[HBD99] K.U. Heitmann, B. Blobel und J. Dudeck. *HL7-Kommunikationsstandard in der Medizin*. Mönch Verlag, 1. Auflage, 1999.

[HW03] G. Hohpe und B. Woolf. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. Addison-Wesley, 1. Auflage, 2003.

[KG01] K.A. Kuhn und D.A. Giuse. From Hospital Information Systems to Health Information Systems. *Method. Inform. Med.*, 4:275–287, 2001.

[Lan] Mirth Connect – Der universelle open source Kommunikationsserver, http://mirthconnect.wordpress.com/.

[QSI] Offizielle Seite von Quality Systems, Inc. (QSI) zu Mirth Connect: http://www.mirthcorp.com/products/mirth-connect.

[Rhi] Rhino, https://developer.mozilla.org/en/docs/Rhino.

[SAP99] SAP AG. *IS-HCM Guide for External System Partners*, 1999.

[Spr07] R. Spronk. Die Rolle eines Kommunikationsservers: Ein Überblick für das Management, http://www.ringholm.de/docs/00100_de.htm. Whitepaper, 2007.