Serviceorientiertes Datenqualitätsmanagement auf Basis von Open Source Tools

Julia Klingner, David Ramón Engelhardt, Holger Hinrichs

Fachbereich Elektrotechnik und Informatik
Fachhochschule Lübeck
Mönkhofer Weg 239
23562 Lübeck
julia.klingner@stud.fh-luebeck.de
david.ramon.engelhardt@stud.fh-luebeck.de
hinrichs@fh-luebeck.de

Abstract: Unternehmen sind heute mehr denn je auf qualitativ hochwertige Datenbestände angewiesen, sei es zur effizienten Abwicklung des Tagesgeschäfts oder zur taktischen und strategischen Entscheidungsunterstützung. Unvollständige, inkonsistente oder redundante Daten stören den Prozessablauf, führen zu Nachbearbeitungsaufwänden und gefährden Managemententscheidungen. Umso wichtiger ist es, Datenqualitätsmängel möglichst frühzeitig zu erkennen. Wendet man den klassischen PDCA-Zyklus nach Deming [De82] auf den Bereich Datenqualität an, müssen zunächst in einer Planungsphase (Plan) aktuelle Qualitätsanforderungen an Daten sowie Messinstrumente spezifiziert werden. Bei der Ausführung von Geschäftsprozessen (Do) wird laufend die Datenqualität gemessen. Die anschließende Analyse von Messergebnissen (Check) führt – je nach Bedarf – zum Ergreifen von Verbesserungsmaßnahmen (Act), z. B. einer Datenbereinigung oder Ursachenbekämpfung.

In dem studentischen Projekt DServ (Serviceorientiertes Datenqualitätsmanagement) wurde eine Softwarelösung konzipiert und implementiert, die insbesondere die Phasen Plan, Do und Check unterstützt. Die Messung der Datenqualität erfolgt dabei über einen Satz von Web Services, die mit dem Framework Apache CXF implementiert sind und über einen Apache Tomcat Server bereitgestellt werden. In einem Planungswerkzeug lassen sich Qualitätsanforderungen beschreiben und mit den Web Services assoziieren. Als Beispielszenario dient ein ETL-Prozess, der mit Pentaho Data Integration (Kettle) erstellt wurde. Im Rahmen von DServ wurde ein Plug-In für Kettle entwickelt, welches es erlaubt, die o. a. Qualitätsplanung (XML) einzulesen, mit dem ETL-Datenstrom zu verknüpfen und dynamisch die in der Planung vorgesehenen Web Services zur Qualitätsmessung aufzurufen. Messergebnisse werden in XML-Dateien abgelegt, die wiederum in ein Dashboard-Werkzeug eingelesen werden. Dort erfolgt eine graphische Aufbereitung, die es dem/der Qualitätsverantwortlichen erlaubt, zeitnah auf visualisierte Datenqualitätsmängel zu reagieren. Die Softwarekomponenten sind in Java geschrieben und verwenden das Binding-Werkzeug JiBX zur Abbildung von XML auf Java-Objekte und umgekehrt.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

1 Hintergrund des Projekts

Viele Organisationen haben massive Probleme mit qualitativ mangelhaften Daten, seien es inkonsistente, veraltete bzw. fehlende Werte oder Dubletten. Die Ursachen dieser Probleme können vielfältig sein, wie Abbildung 1 zeigt. Während die Bedeutung von Daten für die Durchführung von Prozessen und die Beantwortung taktischer und strategischer Fragen immer weiter steigt, nimmt auch der Bedarf an effizienten Lösungen zum Datenqualitätsmanagement (DQM) stetig zu [HGH+11]. In Analogie zur Qualitätssicherung in der industriellen Fertigung erscheint es auch bei datenverarbeitenden Prozessen sinnvoll, an geeigneten Stellen des Prozessablaufs Qualitätsmessungen durchzuführen und aus der Auswertung der Messdaten Konsequenzen zu ziehen, z. B. indem mangelhafte Daten ausgesondert oder als solche markiert werden bzw. indem versucht wird, die Ursachen der DQ-Probleme zu beseitigen. Diesen Ansatz verfolgt das Projekt DServ (Serviceorientiertes Datenqualitätsmanagement), welches im Rahmen der Lehrveranstaltung „Softwaretechnik-Projekt“ im sechsten Semester des Studiengangs Informatik/Softwaretechnik an der Fachhochschule Lübeck durchgeführt wurde. An dem Projekt beteiligten sich zehn Studierende.

Abbildung 1: Ursachen von Datenqualitätsmängeln

Der Beitrag beschreibt Anforderungen an die zu entwickelnde Lösung, skizziert wesentliche Entwurfsentscheidungen und stellt die implementierte Lösung vor. Ein besonderer Fokus liegt dabei auf einer Bewertung der verwendeten Open Source Tools. Ein Ausblick auf anstehende Weiterentwicklungen rundet den Beitrag ab.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

2 Anforderungen und Werkzeugauswahl

Als Anwendungsbeispiel für die zu entwickelnde Softwarelösung wurden ETL-Prozesse (Extract, Transform, Load) zur Befüllung eines Data Warehouse gewählt. ETL-Prozesse integrieren Daten aus heterogenen Quellen, wobei Datenqualitätsmängel leicht übersehen werden oder sogar erst entstehen. Da Data Warehouse-Systeme auf qualitativ hochwertige Daten angewiesen sind, ist der Bedarf an begleitenden Qualitätssicherungsmaßnahmen in solchen Prozessen besonders hoch. Grundsätzlich soll die Software aber auch für andere datenverarbeitende Prozesse einsetzbar sein.

Mit der Software soll es möglich werden, automatisierte Datenqualitätsprüfungen im Kontext von ETL-Prozessen zu konfigurieren, durchzuführen und auszuwerten. Das verwendete ETL-Werkzeug ist für die Bereitstellung der zu prüfenden Daten verantwortlich, so dass eine maximale Flexibilität hinsichtlich der Quellenanbindung erreicht wird.

Die Geschäftsregeln, die Qualitätsanforderungen an die Daten beschreiben, werden auf einen Satz von DQ-Prüfoperationen abgebildet. Jede Prüfoperation erwartet bestimmte Parameter und liefert, nachdem alle verarbeiteten Datensätze geprüft wurden, je nach Erfüllungsgrad der Regel bestimmte Ergebniswerte zurück. Diese lassen auf die Datenqualität schließen und werden im Folgenden als Key Performance Indicators (KPIs) bezeichnet. Für jede DQ-Prüfung wird anhand eines Schwellwerts (Threshold) festgelegt, ab wann ein Messwert als kritisch eingestuft wird. Die KPIs werden visualisiert, wobei kritische Werte hervorgehoben werden. Es ist möglich, die KPIs mehrerer Prüfungen gewichtet zu aggregieren und diesen Aggregaten selbst eigene Schwellwerte zuzuweisen.

Als ETL-Software wurde das quelloffene Werkzeug Pentaho Data Integration (Kettle) [CBD10, Pe11] gewählt. Der enthaltene ETL-Editor Spoon ermöglicht es, den ETL-Prozess grafisch zu modellieren. Hierzu werden einzelne Prozessschritte (Steps) zu komplexen Abläufen verschaltet. Manche Steps dienen der Extraktion von Daten, andere der Transformation und wieder andere der Speicherung der aufbereiteten Daten.

Spoon verfügt über eine Plugin-Architektur, welche es ermöglicht, eigene Steps zu implementieren. Im Rahmen des DServ-Projekts galt es, ein Plugin für Spoon zu entwerfen, über das DQ-Prüfungen nahtlos in einen ETL-Prozess eingebunden werden können. Bei der Ausführung des Prozesses laufen die DQ-Prüfungen so automatisiert mit.

Um die eigentlichen DQ-Prüfungen so flexibel wie möglich zu gestalten, wurde entschieden, diese als Web Services zu implementieren, welche von dem DServ-Plugin mit Werten aus dem Datenstrom aufgerufen werden. Die Web Services sollten mit dem Framework Apache CXF [Ke10, Ap11a] implementiert und über einen Apache Tomcat Server [Ap11b] bereitgestellt werden.

Die Spezifikationen der DQ-Prüfungen sowie die Messergebnisse sollten in einem zu entwickelnden XML-Format gespeichert werden. Als Binding-Werkzeug zur Abbildung von XML auf Java-Objekte und umgekehrt sollte JiBX [So11] zum Einsatz kommen, da mit diesem Werkzeug bereits Projekterfahrungen vorlagen.

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4
weitere Artikel online:
http://informatik2011.de/519.html

Des Weiteren wurden JFreeChart [JFr11] zur Visualisierung von Messergebnissen, JBoss Drools [Bro09, JBo11] zur regelbasierten DQ-Prüfung sowie der MiG Layout Manager [MiG11] zur Gestaltung der Benutzungsoberfläche ausgewählt.

3 Softwaretechnisches Konzept und Implementierung

In diesem Abschnitt werden die vier Komponenten, die im Rahmen des DServ-Projekts entwickelt wurden, näher beschrieben. Einen Überblick über das Zusammenspiel der Komponenten gibt Abbildung 2. Alle DServ-Komponenten sind in Java geschrieben und somit auf allen gängigen Betriebssystemen lauffähig, sofern eine JVM installiert ist.

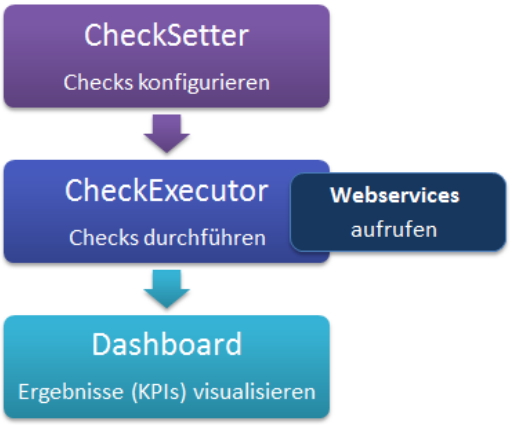

Abbildung 2: Zusammenspiel der DServ-Komponenten

3.1 CheckSetter

Der CheckSetter dient der Konfiguration der Geschäftsregeln bzw. durchzuführenden DQ-Prüfungen. Hierzu wird er mit den zu verwendenden Web Services konfiguriert, von denen er Informationen darüber bezieht, welche DQ-Prüfungen zur Verfügung stehen. Nach dem Anlegen eines neuen Projekts muss dem CheckSetter die URL der WSDL-Datei mindestens eines Web Service mitgeteilt werden, damit die zur Verfügung stehenden Prüfoperationen ausgelesen werden können.

Anschließend werden für jede Stelle im ETL-Prozess, an der Daten geprüft werden sollen, ein Prüfschritt (Step) und die innerhalb dieses Schritts benötigten Datenfelder samt Datentyp spezifiziert. Jeder Step kann beliebig viele Prüfungen enthalten. Zur Konfiguration einer Prüfung wird eine Prüfoperation mit zu prüfenden Datenfeldern und ggf. weiteren Parametern assoziiert. Mehrere Prüfungen können gewichtet zu Aggregaten mit eigenen KPIs zusammengefasst werden.

Darüber hinaus werden im CheckSetter Schwellwerte festgelegt, ab denen die KPIs als kritisch gelten. Eine vollständig spezifizierte Prüfkonfiguration wird als XML-Datei gespeichert, die von den nachgeschalteten DServ-Komponenten (CheckExecutor und Dashboard) eingelesen und interpretiert wird.

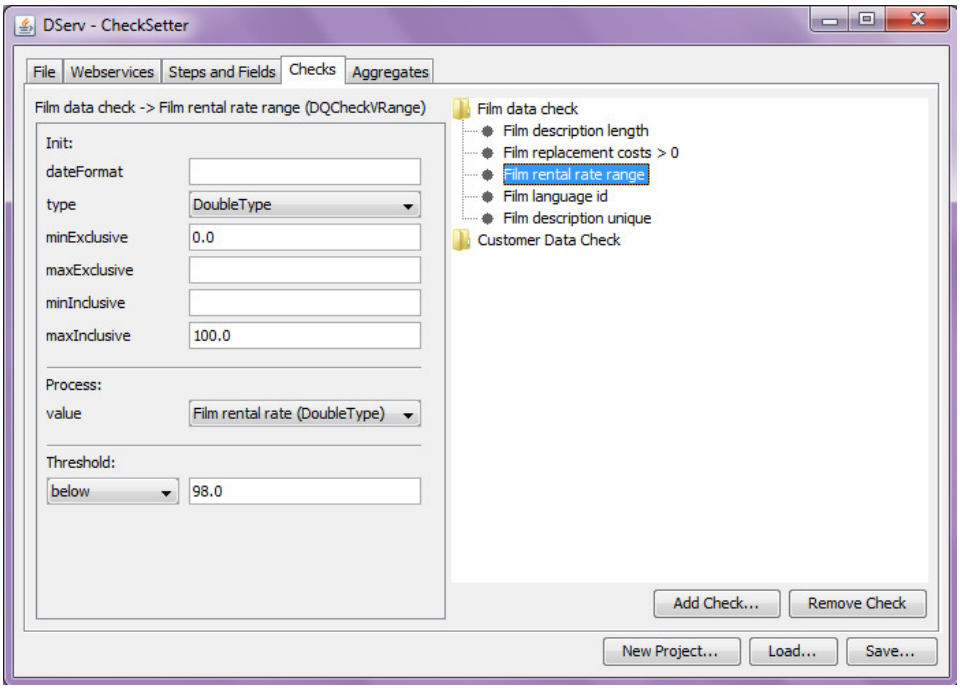

Abbildung 3: Konfiguration einer DQ-Prüfung im CheckSetter

Ein Beispiel für die Konfiguration einer DQ-Prüfung mit dem CheckSetter zeigt Abbildung 3. Die aktuell konfigurierte Prüfung „Film rental rate range“ prüft das Datenfeld „Film rental rate“ vom Typ Double im Step „Film data check“. Ein Messwert wird hier als kritisch eingestuft, wenn er unter den Schwellwert von 98% fällt.

3.2 Web Services

Die im DServ-Projekt entwickelten Web Services bieten DQ-Prüfungen an. Jede Prüfung besteht aus drei Methoden: eine zur Initialisierung der Prüfung (Init), eine zur Prüfung eines Datensatzes (Process, wird wiederholt aufgerufen), und eine zum Beenden des DQ-Prüfungsdurchlaufs und zum Abfragen des ermittelten KPI (Finish).

Die Init-Methode wird mit Parametern aufgerufen, die die Prüfung konfigurieren. Als Parametertypen werden derzeit Boolean, String, Double, Integer und DateTime sowie Arrays dieser Datentypen unterstützt. Des Weiteren können beliebige, im Web Service frei definierbare Enums als Parameter verwendet werden. Die Init-Methode hat keinen Rückgabetyp.

Die Process-Methode wird für jeden durchlaufenden Datensatz einmal mit den zu prüfenden Werten als Parameter aufgerufen. Die Prüfung kann sich auf nur ein oder auf mehrere Datenfelder beziehen, je nach Art der Prüfung. Gültige Parametertypen sind hier Boolean, String, Double, Integer und DateTime. Als Rückgabe ist je nach Art der Prüfung entweder ein boolescher Wert (true, wenn die in der Init-Methode festgelegten Bedingungen erfüllt wurden) oder ein String (mit detaillierten Fehlerbeschreibungen) vorgesehen.

Die Finish-Methode beendet einen Prüfungsdurchlauf. Sie erwartet keine Parameter und gibt einen Double-Wert zwischen 0 und 1 zurück, der angibt, welcher Prozentsatz der Datensätze die Prüfung bestanden hat.

Ein Prüfungsdurchlauf wird mit Hilfe einer HTTP-Session eindeutig identifiziert. Auf diese Weise können verschiedene Prüfungsdurchläufe mit unterschiedlichen Konfigurationen auf denselben Web Service-Methoden parallel durchgeführt werden.

Insgesamt wurden im Rahmen des DServ-Projekts neun verschiedene Prüfoperationen implementiert, im Einzelnen:

- *DQCheckPlausibility* prüft die Datenkonsistenz anhand von Geschäftsregeln.
- *DQCheckVNotNull* prüft das Vorhandensein von Werten.
- *DQCheckVRange* prüft die Einhaltung von Wertebereichen.
- *DQCheckVFormat* prüft Datenformate anhand Regulärer Ausdrücke.
- *DQCheckDate* prüft die Gültigkeit von Datumsangaben.
- *DQCheckVLength* prüft die Länge von Zeichenketten.
- *DQCheckIdentNo* prüft die Gültigkeit von Kreditkartennummern etc.
- *DQCheckReferentialIntegrity* prüft Verknüpfungen zu Referenzdaten.
- *DQCheckVUniqueness* prüft die Eindeutigkeit von Werten.

Zukünftig ist vorgesehen, diesen Satz von Operationen noch zu erweitern, so dass z. B. Prüfungen auf Einhaltung bestimmter Werteverteilungen möglich werden.

3.3 CheckExecutor

Der CheckExecutor ruft die mit dem CheckSetter konfigurierten DQ-Prüfungen aus einem ETL-Prozess heraus auf. Er ist als Plugin des ETL-Modellierungswerkzeugs Spoon realisiert. Die aufzurufenden Prüfoperationen der angebundenen Web Services sowie die an diese zu übergebenden Parameter bezieht der CheckExecutor aus dem Datenstrom des ETL-Prozesses.

Die ermittelten KPIs speichert der CheckExecutor in einem später vom DServ-Dashboard einzulesenden XML-Dokument. Abbildung 4 zeigt, wie das DServ-Plugin an einer geeigneten Stelle eines ETL-Prozesses eingeklinkt wurde. Um die Laufzeit des Transformationsprozess nicht zu beeinträchtigen, wurde das Plugin hier in einem separaten Zweig des Datenstroms platziert. Sollen Prüfergebnisse hingegen Einfluss auf den Fortgang des ETL-Prozesses haben, z. B. durch Aussonderung mangelhafter Daten, müsste die Einbindung des Plugins natürlich in Serie erfolgen.

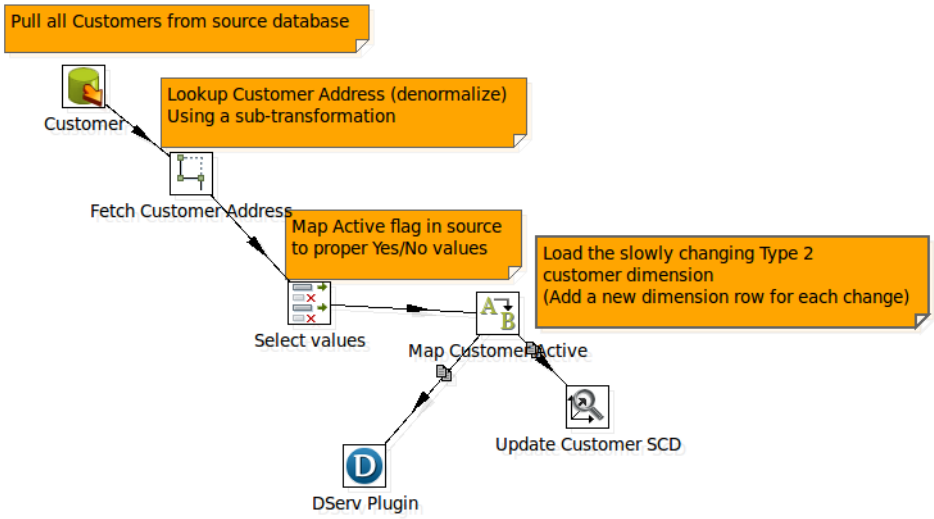

Abbildung 4: ETL-Prozess mit eingeklinktem DServ-Plugin

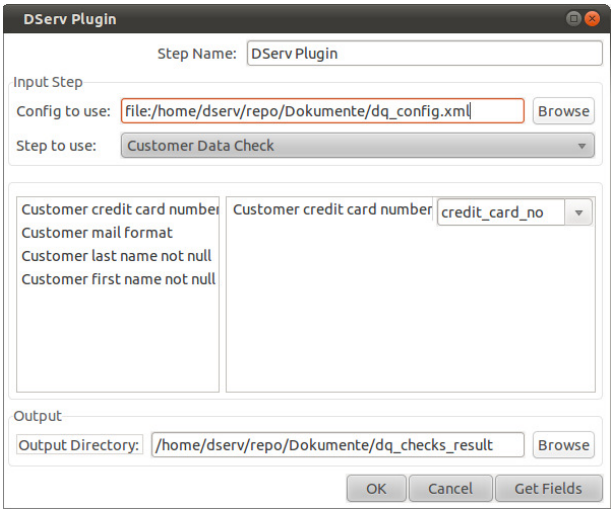

Abbildung 5: Zuordnung von Datenfeldern zu DQ-Prüfungen im CheckExecutor

Ein Doppelklick auf das DServ-Plugin öffnet den Konfigurationsdialog des CheckExecutors. Hier wird eine im CheckSetter erstellte Konfigurationsdatei eingelesen und der für die aktuelle Stelle des ETL-Prozesses passende Step ausgewählt. Die im CheckSetter angegebenen zu prüfenden Felder werden den im Datenstrom zur Verfügung stehenden Feldern zugeordnet, wie in Abbildung 5 dargestellt.

Der CheckExecutor ist die einzige DServ-Komponente, welche an ein konkretes Prozessbeschreibungswerkzeug gebunden ist. Zur Verwendung von DServ mit anderen ETL-Werkzeugen oder Workflowmanagementsystemen müsste diese Komponente entsprechend angepasst werden. Dies kann unabhängig von den anderen DServ-Komponenten geschehen.

3.4 Dashboard

Das DServ-Dashboard visualisiert die ermittelten KPIs. Die Oberfläche ist dabei frei konfigurierbar. Für jeden KPI stehen verschiedene Anzeigeinstrumente zur Auswahl.

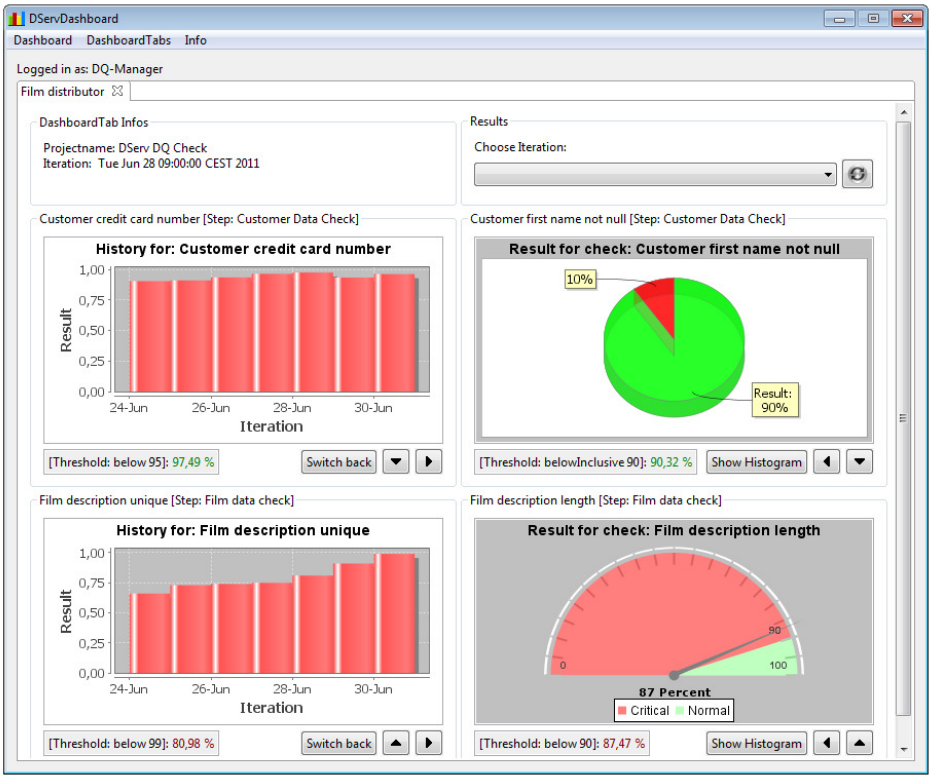

Abbildung 6: Visualisierung von KPIs im DServ-Dashboard

ISBN 978-3-88579-286-4

Dabei können nicht nur die KPIs eines einzelnen Durchlaufs, sondern auch zeitliche Verläufe angezeigt werden. Dadurch lässt sich insbesondere die Wirksamkeit von ursachenorientierten Maßnahmen zur DQ-Verbesserung darstellen. KPIs, welche die im CheckSetter festgelegten kritischen Schwellwerte überschreiten, werden farblich besonders hervorgehoben.

Das Dashboard lässt sich in zwei Modi starten: Im Administrator-Modus, welcher kennwortgeschützt ist, können die Anzeigeinstrumente spezifiziert werden. Der User-Modus hingegen ist auf lesende Zugriffe auf vordefinierte Visualisierungen beschränkt.

Die KPIs werden aus den vom CheckExecutor erstellten Ergebnisdateien ausgelesen. Außerdem werden die kritischen Schwellwerte aus der zugehörigen, im CheckSetter erzeugten Konfigurationsdatei geladen. Die Anzeige der Werte erfolgt über verschiedene Diagrammtypen, wobei je nach Diagrammtyp auch der kritische Schwellwert visualisiert wird. In Abbildung 6 ist zu sehen, wie Prüfergebnisse im DServ-Dashboard dargestellt werden.

4 Bewertung und Ausblick

Dieser Abschnitt fasst Projekterfahrungen mit den verwendeten Open Source-Komponenten zusammen und gibt einen Ausblick auf zukünftige Vorhaben im DServ-Umfeld.

4.1 Bewertung von Apache CXF

Apache CXF [Ke10, Ap11a] unterstützt die Entwicklung von Web Services sowohl über den Code-First- als auch über den Contract-First-Ansatz. Bei Code-First wird zuerst der Java-Quellcode geschrieben und mit Annotationen versehen; anschließend werden daraus die WSDL- und XSD-Dateien generiert. Bei Contract-First hingegen wird zunächst die Schnittstelle per WSDL/XSD festgelegt; später wird diese Schnittstelle in Java implementiert. Bei der Entwicklung einer serviceorientierten Softwarelösung im Team hat sich der Contract-First-Ansatz als geeigneter erwiesen. Werden die Client- und Server-Komponenten getrennt voneinander entwickelt, können beide Subteams parallel arbeiten, zudem kann die Schnittstelle im Vorfeld gemeinsam mit dem Kunden sauber spezifiziert werden.

Die Dynamic Client Facility von Apache CXF erlaubt die Erzeugung von Service Endpoint Interfaces (SEI) und Bean-Klassen zur Laufzeit. Service-Operationen können dynamisch mit passenden Parametern (auch mit komplexen Typen, z. B. Objekten) aufgerufen werden. Sofern die vollqualifizierten Namen der Parametertypen bekannt sind, ist die Instanziierung und Parametrisierung verhältnismäßig einfach. Eine alternative Möglichkeit, die Namen zur Laufzeit zu ermitteln, konnte im Rahmen des Projekts nicht gefunden werden.

Es war daher notwendig, die WSDL-Datei entsprechend zu parsen. Vorhandene Bibliotheken, die diese Arbeit übernehmen könnten, sind relativ undurchsichtig. Außerdem ist die Dokumentation von CXF in Hinblick auf Dynamic Clients sehr dürftig – es existiert lediglich ein Code Listing sowie ein Demoprojekt, welches allerdings keinen tieferen Einblick erlaubt. Als Fazit lässt sich daher feststellen, dass – sofern möglich – statische Clients verwendet werden sollten.

Von der geschilderten Problematik mit dynamischen Clients abgesehen, konnten im DServ-Projekt keine Defizite von Apache CXF zum alternativen, bekannteren Framework Apache Axis2 festgestellt werden.

4.2 Bewertung von JiBX

JiBX [So11] ist ein Open Source Tool zur Abbildung von XML auf Java-Objekte und umgekehrt. Die Einstiegshürde ist niedrig, sofern man eine einfache Java-Klasse speichern möchte. Soll jedoch eine bestehende Java-Klasse auf ein bereits festgelegtes XML-Schema abgebildet werden, muss eine eigene Binding-Konfiguration erstellt werden, was den Einstiegsaufwand erheblich erhöht. Dieser Fall ist während des DServ-Projekts aufgetreten – er wurde insofern umgangen, dass ein Adapter für die bestehenden und von JiBX automatisiert erstellten Klassen geschrieben wurde. Der von JiBX generierte Code, sowohl Java als auch XML, ist sehr einfach zu lesen und zu bearbeiten.

Da zum Projektstart bereits Erfahrungen mit JiBX vorlagen und die Entwicklungszeit auf ein Semester begrenzt war, wurde diesem Werkzeug der Vorzug vor dem JAXB-Standard [Gla11] gegeben. Durch die gewählte Softwarearchitektur ist jedoch grundsätzlich eine Austauschbarkeit gegeben, so dass in späteren DServ-Versionen ein Umstieg z. B. auf JAXB mit geringem Aufwand möglich wäre.

4.3 Bewertung von Pentaho Data Integration (Kettle)

Für das DServ-Projekt wurden ETL-Strecken als Anwendungsbeispiel für datenverarbeitende Prozesse gewählt, die mit DQ-Prüfungen angereichert werden sollen. Das dazu eingesetzte ETL-Werkzeug Pentaho Data Integration [CBD10, Pe11], auch als Kettle bekannt, hat sich dabei als überaus flexibel einsetzbar erwiesen. Dies betrifft zum einen die große Anzahl unterstützter Quell- und Zielformate sowie die umfangreiche Transformationsbibliothek, zum anderen aber auch die Erweiterbarkeit um zusätzliche Plugins, wie im DServ-Projekt praktiziert. Durch die Verwendung von Kettle in Lehrveranstaltungen erhalten Studierende einen guten Eindruck von der Leistungsfähigkeit professioneller Datenintegrationswerkzeuge. Eine spätere Einarbeitung in kommerzielle Werkzeuge von Herstellern wie Oracle und Informatica wird dadurch erheblich erleichtert.

4.4 Bewertung von JBoss Drools

Die Rule Engine JBoss Drools [Bro09, JBo11] wurde im DServ-Projekt nur am Rande bei der Implementierung der Prüfoperation DQCheckPlausibility eingesetzt. Drools verfügt über eine mächtige Sprache zur Modellierung von deskriptiven Geschäftsregeln, die zur Laufzeit auf Java-Objekte angewandt werden. Für die Regelerstellung stehen einfache grafische Editoren zur Verfügung, die jedoch nicht den gesamten Sprachumfang unterstützen. Einfache Regeln für DQ-Prüfung ließen sich damit jedoch problemlos realisieren.

Alternativ ist die Spezifizierung sog. Domain Specific Languages (DSL) möglich, mit denen scheinbar natürlich-sprachliche Regeln erstellt werden können. Leider hat sich dieser interessante Ansatz als noch nicht praxistauglich erwiesen, da man sich bei der Regelerstellung exakt an die vorab spezifizierte Syntax halten muss, was recht fehleranfällig ist. Dass das Debugging auf der ursprünglichen Regelsprache und nicht auf den DSLs erfolgt, erschwert deren Einsatz zusätzlich.

4.5 Bewertung von JFreeChart

Die Verwendung von JFreeChart [JFr11] hat sich für Neueinsteiger als teilweise schwierig herausgestellt, da leider kein offizielles „Getting Started“ zur Verfügung steht, sondern lediglich eine JavaDoc. Es existiert zwar eine Demo-Jar, in der die Diagrammtypen beispielhaft angezeigt werden, den Quellcode dazu gibt es jedoch nur mit dem Developer Guide, welcher für 50 Euro käuflich erworben werden muss. Anwendungsbeispiele mit Quellcode gibt es nur von unabhängigen Anbietern. JFreeChart ist auf AWT aufgebaut, allerdings stehen auch auf SWT basierende Quellen zur Verfügung. Trotz der genannten Widrigkeiten konnten nahezu alle Anforderungen an die Visualisierung von Messergebnissen mit Hilfe von JFreeChart erfüllt werden.

4.6 Bewertung von MiG Layout

MiG Layout [MiG11] ist ein freier Layout-Manager für Swing und SWT. Er ist ausführlich dokumentiert und wird von einer lebendigen Community betreut. Es gibt Bestrebungen, MiG Layout dem offiziellen Java Development Kit (JDK) hinzuzufügen; gegenwärtig steht das Werkzeug auf Platz 3 der Top 25 Requests for Enhancements (RFE) [SDN11] der Java Bug Database.

Ein großer Vorteil von MiG Layout gegenüber anderen (im JDK enthaltenen) Layout-Managern ist die einfache Handhabung und der gut lesbare Code, mit dem Benutzungsoberflächen gestaltet werden. Einfache Layouts sind mittels MiG Layout fast trivial zu programmieren, komplexe Layouts bleiben übersichtlich und gut wartbar. Auch die dynamische Generierung von Benutzungsoberflächen zur Laufzeit ist häufig einfacher zu handhaben als mit anderen Layout-Managern.

4.7 Zukünftige Vorhaben

Die Ergebnisse des DServ-Projekts sollen in weiteren studentischen Lehrveranstaltungen und Abschlussarbeiten aufgegriffen und ausgebaut werden. Derzeit läuft bereits eine Bachelorarbeit, die sich mit der Erkennung von Dubletten in Datenbeständen sowie mit deren semiautomatischer Beseitigung beschäftigt.

Des Weiteren gilt es, einzelne Defizite des aktuellen Prototypen auszumerzen. So sollen z. B. Ergebnisse von DQ-Prüfungen mittelfristig in einer XML-Datenbank und nicht mehr im Dateisystem abgelegt werden.

Wünschenswert wäre es, in Zusammenarbeit mit einem Wirtschaftsunternehmen ein Pilotprojekt durchzuführen, um praktische Erfahrungen mit dem Einsatz von DServ in realen Geschäftsprozessen und mit Echtdaten zu sammeln.

Derzeit gibt es keine Planungen, das Ergebnis des DServ-Projekts selbst als Open Source zu veröffentlichen. Der Hauptgrund hierfür ist, dass die Software von einer großen Gruppe von Studierenden mit unterschiedlichen Fähigkeiten entwickelt wurde. Der Code wurde somit recht heterogen und müsste zunächst aufwändig restrukturiert werden, bevor er guten Gewissens Open Source gestellt werden kann.

Quellenverzeichnis

[Ap11a] Apache Software Foundation: Apache CXF Website, http://cxf.apache.org/, 2011.
[Ap11b] Apache Software Foundation: Apache Tomcat Website http://tomcat.apache.org/, 2011.
[Bro09] Browne, P.: JBoss Drools Business Rules. Packt Publishing, 2009.
[CBD10] Casters, M.; Bouman, R.; van Dongen, J.: Pentaho Kettle Solutions: Building Open Source ETL Solutions with Pentaho Data Integration. John Wiley & Sons, 2010.
[De82] Deming, W. E.: Out of the Crisis. MIT Press, 1982.
[Gla11] Glassfish: JAXB Website, http://jaxb.java.net/, 2011.
[HGH+11] Hildebrand, K.; Gebauer, M.; Hinrichs, H.; Mielke, M.: Daten- und Informationsqualität – Auf dem Weg zur Information Excellence. 2. Auflage, Vieweg & Teubner, 2011.
[JBo11] JBoss: JBoss Drools Website, http://www.jboss.org/drools/, 2011.
[JFr11] JFree: JFreeChart Website, http://www.jfree.org/jfreechart/, 2011.
[Ke10] Kent, K. I. T.: Developing Web Services with Apache CXF and Axis2. 3. Auflage, Verlag lulu.com, 2010.
[MiG11] MiG: MiGLayout Website, http://www.miglayout.com/, 2011.
[Pe11] Pentaho Corporation: Pentaho Kettle Website, http://kettle.pentaho.com/, 2011.
[SDN11] Sun Developer Network: Top 25 Requests for Enhancements, http://bugs.sun.com/top25_rfes.do, 2011.
[So11] Sourceforge: JiBX Website, http://jibx.sourceforge.net/, 2011.