Institut für Parallele und Verteilte Systeme
Abteilung Anwendersoftware
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Fachstudie Nr. 107

Evaluation von Datenintegrationssystemen im e-Science-Bereich

Daniel Beck Jens Müller Frank Ruthardt

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr.-Ing. habil. Bernhard Mitschang
Betreuer:	Dipl.-Inf. Peter Reimann
begonnen am:	5. Mai 2009
beendet am:	30. September 2009
CR-Klassifikation:	H.2.5, H.2.8

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Datenintegrationssysteme (*DIS*) zielen darauf ab, Anwendungen einen einheitlichen Zugriff auf verschiedene heterogene Datenquellen (relationale Datenbanksysteme, Webservices, etc.) bereitzustellen, welche für sich jedoch weitgehend autonom bleiben. Sie bilden dabei meist eine zusätzliche Schicht zwischen den Datenquellen und den Anwendungen.

1.1 Motivation

Die wissenschaftliche Arbeit erfordert Anwendungen zur Verarbeitung sehr großer und heterogener Datenmengen. Datenintegrationssysteme können hierbei genutzt werden, um Daten von mehreren Datenquellen zu kombinieren und erleichtern dem Wissenschaftler dadurch den Zugriff auf die Datenquellen. Dies soll dazu führen, dass Wissenschaftler eher spezifizieren, welche Informationen sie benötigen, anstatt, wie sie diese Informationen bekommen.

Weitere Vorteile von Dantenintegrationssystemen sind, dass den auf das Datenintegrationssystem zugreifenden Anwendungen oft auch bei Änderungen der lokalen Datenschemata und der Datenquellen eine einheitliche und weitgehend stabile Schnittstelle geboten werden kann. Im verteilten Szenario können durch Datenintegrationssysteme Datentransformationen näher an den Datenquellen durchgeführt werden. Daraus resultiert in der Regel ein Effizienzvorteil im Vergleich zum Szenario, bei dem alle Daten erst bei der Anwendung vollständig verarbeitet werden.

Die Abteilung Anwendersoftware am Institut für Parallele und Verteilte Systeme nimmt an der Forschung im Rahmen des Exzellenz-Clusters *Simulation Technology* (*SimTech*) der Universität Stuttgart teil. Das Teilprojekt *Data provisioning for scientific workflows* des Exzellenz-Clusters ist Teil des übergeordneten Projektnetzwerks *Integrated data management, workflow and visualization to enable an integrative systems science*. Aufgabe des Projektnetzwerks ist die Untersuchung und Entwicklung neuartiger Methoden und Algorithmen zur Modellierung, Ausführung und Verwaltung von Simulationsworkflows. Im Rahmen des Teilprojekts *Data provisioning for scientific workflows* wird eine konsolidierte Datenmanagement- und Datenbereitstellungsabstraktion bereitgestellt, die effizient über Workflow-Methoden und neue Visualisierungstechniken in Simulationsberechnungen angesteuert werden kann. Dazu können Datenintegrationssysteme verwendet werden.

Diese Fachstudie evaluiert einige der am Markt verfügbaren Datenintegrationssysteme und soll dem Forschungsprojekt als Entscheidungshilfe zur Auswahl, Verwendung und möglichen Erweiterung eines der evaluierten Systeme dienen. Die Anforderungen, die dabei

an die Systeme gestellt werden, werden im Abschnitt 8.2 beschrieben. Sollten sich alle evaluierten Systeme als ungeeignet herausstellen, dokumentiert die vorliegende Arbeit die Anforderungen an eine dadurch notwendige Eigenentwicklung.

1.2 Aufbau dieses Dokuments

Die vorliegende Arbeit ist in folgende Abschnitte gegliedert:

Kapitel 2 – Schema zur Klassifikation und Evaluation von Datenintegrationssystemen: In diesem Kapitel werden die Kriterien, nach denen die untersuchten Systeme klassifiziert und evaluiert werden, genannt und beschrieben.

Kapitel 3 – Untersuchte Systeme: Dieses Kapitel gibt einen Überblick über die für diese Arbeit evaluierten Systeme. Die vier vielversprechendsten Systeme werden in den Kapiteln 4-7 detailliert ensprechend des Evaluationsschemas aus Kapitel 2 evaluiert.

Kapitel 4 – Evaluation von OGSA-DAI: In diesem Kapitel werden die Ergebnisse der Evaluation der Software *Open Grid Service Architecture Data Access and Integration* (*OGSA-DAI*) anhand des Evaluationsschemas nach Kapitel 2 beschrieben.

Kapitel 5 – Evaluation von BioMart: Dieses Kapitel beschreibt die Ergebnisse der Evaluation des Systems *BioMart*.

Kapitel 6 – Evaluation von Altova MapForce: Dieses Kapitel beschreibt die Ergebnisse der Evaluation von *Altova MapForce*. *Altova MapForce* ist das erste der beiden in dieser Arbeit betrachteten kommerziellen Produkte.

Kapitel 7 – Evaluation von DataDirect Data Integration Suite: Dieses Kapitel beschreibt die Ergebnisse der Evaluation der *DataDirect Data Integration Suite*. *DataDirect Data Integration Suite* ist das zweite der beiden in dieser Arbeit betrachteten kommerziellen Produkte.

Kapitel 8 – Zusammenfassung und Bewertung: In diesem Kapitel werden die Ergebnisse der Arbeit zusammengefasst und die Systeme abschließend hinsichtlich der möglichen Nutzung im in Kapitel 1.1 genannten Anwendungsgebiet verglichen.

Haftungsausschluss

Die Autoren haben das Dokument mit größter Sorgfalt erstellt und sich auf die referenzierte Literatur gestützt. Andere Informationen bzw. Quellen wurden nicht benutzt. Dennoch können Fehler und Unklarheiten nicht vollständig ausgeschlossen werden. Die Autoren übernehmen deshalb keine Gewähr für die Aktualität, Richtigkeit und Vollständigkeit der Inhalte.

2 Schema zur Klassifikation und Evaluation von Datenintegrationssystemen

Theoretische Ergebnisse zur Untersuchung und Einordnung von Datenintegrationssystemen liefern [Rah94] , [BKLW99] und [Reio8]. Einige der folgenden Klassifikations- und Evaluationskriterien, denen die untersuchten Datenintegrationssysteme in den Kapiteln 4, 5, 6 und 7 unterzogen wurden, basieren hierauf. Die übrigen Kriterien wurden während der Fachstudie erarbeitet.

2.1 Klassifikationen von Datenintegrationssystemen

Die Klassifikationen unterteilen die Datenintegrationssysteme in unterschiedliche grundlegende Gruppen. Die Unterteilung betrifft insbesondere die Art der Integration sowie das Existieren einer materialisierten Datenbasis. Darüber hinaus gibt es eine Vielzahl weiterer Kriterien, anhand derer die Datenintegrationssysteme eingeordnet werden können. Für die Evaluation in dieser Arbeit werden jedoch nur die genannten Klassifikationen berücksichtigt.

2.1.1 Enge oder lose Integration

Bei Datenintegrationssystemen manifestiert sich die Integrationsart dadurch, ob die Anwendungen, die das Datenintegrationssystem nutzen, die lokalen Schemata der integrierten Datenquellen sehen, oder diese durch ein globales Schema verborgen sind.

Bei der **engen Integration** gibt es *ein* integriertes Zugriffsschema, auf das alle Benutzer des Datenintegrationssystems zugreifen. Dabei muss dieses globale Schema die Semantiken aller integrierten lokalen Schemata auf eine gemeinsame Basis bringen. Das globale Schema kann manuell, semiautomatisch oder durch automatische Schemaintegration erstellt worden sein. Bei der engen Integration ist es häufig schwierig, Heterogenitäten der verschiedenen Quellschemata aufzulösen. Dies bezieht sich sowohl auf syntaktische Heterogenitäten, also solche, die aufgrund struktureller Unterschiede der Quellschemata vorherrschen, als auch semantische Heterogenitäten, die aufgrund von Bedeutungsunterschieden der Quellschemata gegeben sind. Diese Heterogenitäten müssen aufgelöst werden, um eine sinnvolle Integration zu ermöglichen.

Die **lose Integration** bietet typischerweise nur eine einheitliche Anfragesprache an, die die Heterogenität der unterschiedlichen Anfragesprachen der unterliegenden Datenquellen

versteckt. Dazu muss das Datenintegrationssystem die lokalen Schemata nach außen sichtbar machen. Jede auf das Datenintegrationssystem zugreifende Anwendung ist aber selbst dafür verantwortlich, alle weiteren Heterogenitäten aufzulösen, wobei die globale Anfragesprache häufig Hilfsmittel dafür zur Verfügung stellt. Anwendungen können hier oft auch *integrierte Sichten* definieren, d.h. über verschiedene lokale Schemata kombinierte Sichten, welche unter Umständen auch anderen Anwendungen zur Verfügung gestellt werden können.

2.1.2 Virtuelle oder materialisierte Integration

Eine weitere Klassifikation unterscheidet, ob das Datenintegrationssystem selbst die integrierten Daten persistent speichert. Bei einer virtuellen Integration werden diese bei der Bearbeitung einer Anfrage nur so lange im Speicher (Cache oder Hauptspeicher) gehalten, bis die gesamte Anfrage abgearbeitet ist. Bei einer wiederholten Anfrage, müssen die Daten erneut aus den Datenquellen geladen und integriert werden. Bei einer materialisierten Integration werden die integrierten Daten persistent im Speicher gehalten. Dadurch können Daten aufbereitet und schneller wiederholt durchforstet werden. Die ständige Aktualisierung der Daten erfordert jedoch einen hohen Aufwand, der insbesondere dann nicht gerechtfertigt ist, wenn selten darauf zugegriffen wird. In manchen Systemen wird eine Mischform aus virtueller und materialisierter Integration verwendet (hybrider Ansatz). D.h. es werden manche Teile, die z.B. sehr häufig benötigt werden materialisiert, während andere Teile nicht materialisiert werden.

2.2 Evaluationsschema nach gebotener Funktionalität

Neben den genannten Klassifikationskriterien werden die Datenintegrationssysteme auf die in den nachfolgenden Abschnitten beschriebenen Merkmale bezüglich ihrer gebotenen Funktionalität untersucht.

2.2.1 Erweiterbarkeit und Änderbarkeit

Eine wichtige Anforderung an Datenintegrationssysteme ist die Möglichkeit der effizienten Erweiterung und Anpassung an geänderte Rahmenbedingungen. Sowohl das Hinzufügen, strukturelle Ändern und Entfernen von lokalen Informationssystemen und deren Datenschemata, als auch das Anbieten neuer Dienste und die Änderung der Abhängigkeiten zwischen den integrierten Daten und den zu integrierenden Daten sollte möglich sein. Eine lose Kopplung der lokalen Informationssysteme an das DIS, zum Beispiel über auswechselbare Module, erleichtert dies, während bei enger Kopplung der Austausch eines lokalen Informationssystems erschwert ist. Weiterhin ist es bei Open-Source-Systemen auch möglich, Fehler oder Unzulänglichkeiten in der Implementierung selbst zu beheben.

2.2.2 Lesender oder schreibender Zugriff auf lokale Datenquellen

Viele Datenintegrationssysteme bieten nur lesenden Zugriff auf die lokalen Informationssysteme. Die Probleme beim Schreibzugriff über das Datenintegrationssystem resultieren v.a. daraus, dass viele Quellen überhaupt keinen Schreibzugriff anbieten (z.B. Webseiten als Quellen), dass die Semantik für das Schreiben unklar ist (z.B. bei mehrfacher Existenz von semantisch äquivalenten Objekten in verschiedenen Quellen) sowie dass bei integrierten Sichten mindestens die selben Probleme auftreten, die bereits beim Schreiben auf Sichten bei herkömmlichen (nicht-integrierten) Datenbanksystemen existieren (Kapitel 2.12 aus [KE97]).

2.2.3 Unterstützte Zugriffsmethoden

Es wird hier die Art des Zugriffs einer Anwendung auf das DIS oder auf die resultierenden Daten des DIS untersucht. Dabei sind folgende Arten des Zugriffs möglich:

Per Anfragesprache: Dabei können von der Anwendung frei definierbare Anfragen (z.B. SQL-Anfragen) über eine native Schnittstelle oder einen Datenbanktreiber (ODBC/JDBC) an das DIS übergeben werden.

Über eine vordefinierte parameterisierte Funktion: Dies sind vordefinierte Anfragen, bei denen einige Positionen variabel sind (z.B. Stored Procedures).

Durch Navigation: Es wird nur eine Navigation durch die Graphstruktur der Daten angeboten, wie z.B. bei www-Seiten als Zugriffsquelle.

2.2.4 Integrierbare Zugriffsquellen

In diesem Abschnitt wird untersucht, welche lokalen Quellen (lesend und evtl. schreibend) das Datenintegrationssystem integrieren bzw. ansprechen kann. Dies können u.a. alle Arten von Datenbanken sowie Dateiquellen, Webseiten bzw. Webservices sein.

2.2.5 Autonomie der lokalen Datenquellen

Datenquellen verlieren im Verbund mit anderen Systemen unweigerlich Autonomie, verglichen mit einem alleinstehenden System. In diesem Evaluationsschritt wird deshalb untersucht, welche Arten von Autonomie zu welchem Grad erhalten bleiben. Dabei wird zwischen folgende Arten der Autonomie unterschieden:

Ausführungsautonomie: Inwiefern sind die lokalen Informationssysteme autonom hinsichtlich der Ausführung einer lokalen Anfrage? Beispielsweise muss im Falle einer globalen Transaktion mit einem Commit-Protokoll die lokale (Teil-)Anfrage mit anderen Systemen koordiniert werden.

Kommunikationsautonomie: Kann ein lokales Informationssystem selbst entscheiden, zu welchem Ausmaß und mit welchen anderen Informationssystemen oder mit dem Datenintegrationssystem es kommuniziert?

Entwurfsautonomie: Kann ein lokales Informationssystem autonom festlegen, welche Miniwelt es wie darstellt, sowohl semantisch als auch strukturell?

2.2.6 Aufzulösende Heterogenitäten

Datenquellen können auf vielfältige Weise heterogen sein. Für diese Fachstudie wird die Heterogenität auf folgende Weise aufgeteilt: Heterogenität bezüglich

Des Datenmodells: Hierbei wird unterschieden, ob es sich z.B. um ein relationales, satzorientiertes, hierarchisches, objektorientiertes oder objektrelationales Modell handelt.

Der angebotenen Zugriffssprache: Einige mögliche Beispiele sind SQL, OQL, XPath oder eine boole'sche Anfrage. Von der selben Zugriffssprache gibt es weiterhin Dialekte oder Unterschiede in der von der Datenquelle unterstützten Mächtigkeit, die vom Datenintegrationssystem aufgelöst werden müssen. Z.B. könnte ein relationales Datenbanksystem keine Sortierung (ORDER BY) unterstützen. Die Sortierung müsste dann durch das Datenintegrationssystem durchgeführt werden. Dabei ist in der Regel ein Effizienzverlust unvermeidbar.

Der Syntax: Daten können bei gleichen Datenmodellen unterschiedlich strukturiert dargestellt werden. In relationalen Datenbanken beispielsweise kann der selbe Sachverhalt in vielen unterschiedlichen Schemata dargestellt werden, in einer Tabelle oder mehreren Tabellen verteilt sein oder unterschiedliche Datentypen nutzen.

Der Semantik: Auch Schemata mit gleicher Struktur können unterschiedliche Bedeutungen haben. Es kann einmal zu *Namenskonflikten* kommen, z.B. wenn Synonyme für semantisch äquivalente Objekte oder Homonyme für semantisch unterschiedliche Objekte in verschiedenen Datenquellen verwendet werden. *Datenkonflikte* können trotz Verwendung des selben Datentyps durch unterschiedliche Darstellung von Objekten entstehen. Dies kann z.B. durch Preise in unterschiedlicher Währung und Dezimaltrennung, Adressen mit unterschiedlicher Formatierung usw. passieren.

Datenintegrationssysteme können diese unterschiedlichen Heterogenitäten auf unterschiedliche Weise auflösen bzw. vor der Anwendung verbergen:

Automatisch: Die Heterogenitäten werden ohne Eingriff durch den Benutzer bzw. die Anwendung aufgelöst.

Semi-automatisch: Nach einer automatischen Auflösung bzw. einem automatischen Auflösungsvorschlag ist ein manueller Eingriff möglich oder notwendig.

Manuell: Die Anwendung bzw. der Benutzer muss die Heterogenitäten selbst auflösen.

Üblicherweise bedingt eine größere Autonomie eine weniger automatische Auflösung von Heterogenitäten. Dabei lässt sich insbesondere ein Zusammenhang zwischen der Entwurfsautonomie und der Heterogenität bezüglich Syntax und Semantik feststellen.

2.2.7 Globale Zugriffsschicht

In diesem Abschnitt wird untersucht, ob eine globale Zugriffsschicht vorhanden ist. Dazu verwenden wir zunächst die folgende Definition einer globalen Zugriffsschicht aus [BKLW99]:

„Application and users access a set of heterogeneous data sources through a federation layer which is a software component that offers a uniform way to access the data stored in data sources. The uniformity is reached with a specific interoperation strategy, e.g. this layer can offer a federated schema, a uniform query language, or a uniform set of source and content descriptions as metadata sets. The data sources usually are integrated into the infrastructure with wrappers which resolves some technical differences.“

Für unsere Betrachtung relevant ist im Wesentlichen die Tatsache, dass für eine globale Zugriffsschicht eine spezielle Schicht im Datenintegrationssystem enthalten sein muss, die einen möglichst einheitlichen, transparenten Zugriff auf alle integrierten Daten der lokalen Datenquellen bereitstellt. Dies kann u.a. durch ein globales Schema oder eine einheitliche Anfragesprache erreicht werden.

Es ist hierbei zu beachten, dass es sich u.a. bei Workflows, die die Daten der Quellen zunächst sammeln und aufbereiten, nicht um eine globale Zugriffsschicht handelt: Eine solche Zugriffsschicht greift auf Basis einer Anfrage auf die Datenquellen zu, wohingegen ein Workflow zunächst alle Daten aggregiert, um dann ohne Beteiligung der Datenquellen die Anfrage zu beantworten.

Ist eine globale Zugriffsschicht vorhanden, lassen sich eine Menge von weiteren Kriterien untersuchen, die diese Schicht betreffen. Dazu gehören u.a. die Sprache, mit der Anfragen an die globale Zugriffsschicht gestellt werden können, ob die globale Zugriffsschicht globale Transaktionen ermöglicht und welche Eigenschaften diese globalen Transaktionen erfüllen, insbesondere ob sie die ACID-Eigenschaften erfüllen. Außerdem lässt sich untersuchen, ob es einen globalen Optimierer gibt, der durch genügend Informationen über die lokalen Systeme in der Lage ist, auf globaler Ebene Anfrageoptimierung durchzuführen. Falls es einen solchen globalen Anfrageoptimierer gibt, bleibt außerdem zu betrachten, wie gut dieser optimierte Ablaufpläne erstellen kann, d.h. insbesondere wie gut die Kosten für die lokalen Sub-Anfragen abgeschätzt werden können und in welchem Umfang diese Kosten sich auf die globale Optimierung auswirken.

2.2.8 Anbindung an Workflow-Systeme oder/und Webservices

Dieser Punkt umfasst sowohl, ob das Datenintegrationssystem als Teil eines Workflows und/oder als Webservice angesprochen werden kann, als auch, ob ein Webservice als Datenquelle für das Datenintegrationssystem dienen kann.

2.3 Sonstige Anforderungen

Im folgenden Abschnitt werden weitere Anforderungen und Qualitätsmerkmale beschrieben. Dies umfasst insbesondere nichtfunktionale Merkmale.

2.3.1 Status des Projekts

Wichtig für den möglichen zukünfitigen Einsatz eines Systems ist der Status des Projekts, also ob am Projekt noch weiter gearbeitet wird oder die Software weiter entwickelt und möglicherweise an neue Technologien angepasst wird und ob Fehler korrigiert werden. Hierzu untersuchen wir insbesondere, wann das letzte Release veröffentlicht wurde, ob neue Versionen angekündigt wurden und ob die Finanzierung des Projekts gesichert ist.

2.3.2 Einsatzbereich

In diesem Bereich wird geprüft, für welchen wissenschaftlichen Einsatzbereich das untersuchte System angedacht ist und wodurch es sich hierfür besonders eignet.

2.3.3 Verbreitung

Neben dem vom Hersteller angedachten Einsatzbereich soll hier herausgefunden werden, in welchen Bereichen das System mit Erfolg eingesetzt wird und wie weit verbreitet dieses System im Einsatz tatsächlich ist. Dazu gehört, sofern dies bei den untersuchten Systemen ermittelt werden kann, wie oft das System eingesetzt wird.

2.3.4 Kaufpreis/Lizenz

Hierbei wird der Kaufpreis des Systems in Erfahrung gebracht und untersucht, unter welcher Lizenz (Beispiele sind verschiedene Open Source Lizenzen, Lizenzen ohne Einschränkung der Anzahl der Benutzer oder Lizenzen pro Benutzer) ein System angeboten wird. Insbesondere wird darauf geachtet, ob es bei kommerziellen Produkten Vergünstigungen für den akademischen Einsatz und die wissenschaftliche Nutzung an der Universität Stuttgart gibt.

2.3.5 Unterstützte Plattformen

Dieser Abschnitt informiert, auf welchen Plattformen das untersuchte System lauffähig ist.

2.3.6 Installation und Administration

Wichtig für die anfängliche Akzeptanz eines Systems im Einsatz ist, wie komplex und aufwendig die Installation und die Administration des Systems ist. Hierzu gehören z.B. die Existenz oder das Fehlen einer automatischen Installationsroutine, die Komplexität der manuellen Installation sowie die Betrachtung der Installationsanleitung.

2.3.7 Benutzungsfreundlichkeit

Hier wird geprüft, inwiefern durch Bedienung, Hilfeunterstützung und Konsistenz eine Benutzungsfreundlichkeit des Systems gegeben ist.

2.3.8 Verfügbarkeit von Hilfestellungen

Für die Anwender des Systems unabdingbar ist die Verfügbarkeit von Hilfestellungen bei Problemen. Dazu gehört v.a., ob die Dokumentation vorhanden, vollständig, effektiv und ausgereift ist und ob es Support vom Hersteller und/oder Unterstützung durch Community-Foren oder eine Mailingliste gibt. Weitere Pluspunkte sind zusätzlich angebotene Hilfestellungen wie Schulungen, Erklärungsvideos oder ähnliches.

2.3.9 Performanz

Da im Anwendungsbereich sehr große Datenmengen verarbeitet werden sollen, ist die Performanz des Datenintegrationssystems verglichen mit dem direkten Zugriff auf die Datenquellen ein wichtiges Kriterium. Durch die Heterogenität der untersuchten Systeme ist eine quantitativer Vergleich dieses Kriteriums jedoch nicht möglich. Deshalb werden hier nur mögliche aufgefallene Besonderheiten bezüglich der Performanz der untersuchten Systeme erwähnt.

3 Untersuchte Systeme

Dieses Kapitel stellt die im Bereich der wissenschaftlichen Datenintegration gefundenen Systeme grob vor. Von diesen Systemen werden in den späteren Kapiteln dieses Berichts die Ergebnisse der Evaluation von OGSA-DAI (siehe Kapitel 4), BioMart (siehe Kapitel 5), Altova MapForce (siehe Kapitel 6) und DataDirect Data Integration Suite (siehe Kapitel 7) genauer beschrieben.

3.1 AutoMed

AutoMed [Aut] ist ein mediatorbasiertes Datenintegrationssystem. Es versucht, die Nachteile von *Global-as-view* und *Local-as-view* zu umgehen. Bei *Global-as-view* wird das globale Schema als Sichten über die lokalen Schemata der Datenquellen ausgedrückt. Nachteilig ist hierbei, dass das globale Schema bei Änderungen eines lokalen Schemas möglicherweise verändert werden muss. Bei *Local-as-view* wird das globale Schema definiert, und die lokalen Schema als Sichten auf diesem formuliert. Dies umgeht das Problem, dass bei der Veränderung eines lokalen Schemas das globale Schema ebenfalls geändert werden muss, bringt jedoch Nachteile hinsichtlich der dadurch notwendigen Transformation von Anfragen. Der in AutoMed implementierte Ansatz wird von den Autoren als *Both-as-view* bezeichnet ([JTMP04]), d.h. sowohl globales Schema als auch die lokalen Schemata sind als *Sicht* des jeweils anderen definiert.

AutoMed umfasst eine Programmierschnittstelle in Java und grafische Werkzeuge. Zu AutoMed wurden bis 2008 mehrere Publikationen verfasst. Die Entwicklung steht seit Anfang 2007 still, die letzte stabile Programmversion wurde im Januar 2006 veröffentlicht. Es ist davon auszugehen, dass diese Software aktuell nicht weiterentwickelt wird, weshalb von einer genaueren Betrachtung abgesehen wird.

3.2 BioMart

BioMart [Biob] ist ein primär in der Bioinformatik eingesetztes System, das einen vereinheitlichten Zugriff auf eine Vielzahl biologischer Datenbanken über das frei zugängliche *BioMart Central Portal* bietet. BioMart ist prinzipiell für jede Art von Daten verwendbar, und alle Software-Komponenten stehen, einschließlich Quellcode, zum freien Download bereit, so dass BioMart selbst installiert und beliebig verändert werden kann.

BioMart erfordert die Erzeugung sogenannter *Marts*, zur Beschreibung von Datenquellen. Ein BioMart-System kann dann auf verschiedene *Marts* bzw. andere BioMart-Systeme zugreifen, um deren Daten zu integrieren.

Die Ergebnisse der Evaluation von BioMart nach dem Evaluationsschema finden sich in Kapitel 5.

3.3 BioMediator

BioMediator [Bioe] war ein Projekt der University of Washington für die Entwicklung eines mehrschichtigen Systems zur allgemeinen Datenintegration. Es ist originär für den Bioinformatikbereich ausgelegt, jedoch könnte man eigene Datenwrapper für andere lokale Systeme hinzufügen.

Das System ist nicht mehr aktuell und wurde seit dem Jahr 2007 nicht mehr weiter entwickelt. Der Installationsversuch verlief reibungslos und sehr einfach, jedoch sind alle Standarddatenquellen, also die Proteindatenbanken im Internet, mit denen die Beispiele arbeiten, nicht mehr verfügbar. Die entsprechenden Server scheinen nicht mehr am Netz zu sein. Als Dokumentation steht nur eine kleine Marketing-Seite zur Verfügung, ansonsten besteht die technische Dokumentation aus Einträgen in der Art „ist noch in Bearbeitung“. Durch das Fehlen der Standarddatenquellen und der fehlenden Dokumentation ist ein Betrieb des Systems nicht gelungen.

In Anbetracht dieser Tatsachen und der nicht mehr aktiven Weiterentwicklung des Systems ist von einer Nutzung abzusehen. BioMediator wird deshalb nicht genauer betrachtet.

3.4 BioSQL

Ziel des BioSQL-Projekts [Bioi] ist es, ein generisches relationales Schema für die Speicherung von molekularbiologischen Daten zu schaffen, um Interoperabilität zwischen den bestehenden Projekten BioPerl [Biof], BioPython [Biog], BioJava [Bioa] und BioRuby [Bioh] zu ermöglichen. Dabei wird für jedes dieser Projekte eine objektrelationale Abbildung zu BioSQL zur Verfügung gestellt. Die Daten selbst können dann in relationalen Datenbanken persistent gespeichert werden. Innerhalb des Pakets werden auch Tools zur Datenintegration mitgeliefert, die biologische Daten parsen, analysieren und transformieren können (z.B. DNA-Kodon in Nukleinsäuren).

Eine Nutzung von BioSQL und dessen Tools könnte für Wissenschaftler im biologischen Bereich interessant sein, die bereits eines der Produkte BioPerl, BioPython, BioJava oder BioRuby im Einsatz haben. So ist es leichter relational auf die biologischen Daten zuzugreifen.

BioSQL wird, da es sich lediglich um ein Schema und kein ganzheitliches System handelt, und somit von Produkten aus dem rein biologischen Einsatzbereich abhängig ist, nicht genauer betrachtet.

3.5 DataDirect Data Integration Suite

DataDirect Data Integration Suite [Datb] ist eine kommerzielle Datenintegrationslösung, die auf dem *XQuery*-Produkt des Herstellers aufbaut. DataDirect XQuery kann relationale Datenbanken, Webservices und XML-Datenquellen direkt und beliebige Textquellen mittels *XML Converters* genannter Komponenten abfragen und entsprechend eines Ziel-XML-Schemas integrieren. Applikationen können von Java, .NET oder von einem Webservice-Client aus auf die zugehörigen Bibliotheken zugreifen.

Die Data Integration Suite umfasst ebenfalls die grafische XQuery-IDE Stylus Studio, mit der in XQuery ausgedrückte Datenintegrationen grafisch bearbeitet und getestet werden können. Der Hersteller bietet umfangreiche Dokumentation zum Produkt. Die Software ist nur für Windows verfügbar, Preise werden auf der unübersichtlichen Hersteller-Website nicht genannt, sie sind erst auf Anfrage erhältlich.

Die Data Integration Suite wird in Kapitel 7 detailliert betrachtet.

3.6 EMBOSS

The European Molecular Biology Open Software Suite (kurz EMBOSS) [Emb] ist ein Software-System für die Molekularbiologie. Es unterstützt eine Vielzahl verschiedener Datenformate und kann transparent auf, via Internet verfügbare, offene Datenbanken zugreifen. EMBOSS wird mit jährlichen *Major Releases* konsequent weiterentwickelt, ist quelloffen und kostenlos verfügbar und bietet mit *Jemboss* eine die Bedienung erleichternde und auf Java basierende Benutzeroberfläche.

Die zahlreichen vorhandenen Werkzeuge sind vollständig auf die Arbeit in der Molekularbiologie ausgerichtet. Über ein Toolkit können zwar weitere Komponenten entwickelt werden, die starke Ausrichtung der Software auf dieses Einsatzgebiet führt jedoch dazu, dass dieses System in der vorliegenden Arbeit nicht näher betrachtet wird.

3.7 Altova MapForce

Altova MapForce [Map] ist ein kommerzielles Datenintegrationssystem, das durch den Hersteller nicht betont für den wissenschaftlichen Bereich ausgelegt ist. Aufgrund seines Funktionsumfang, der den anderen Systemen ähnelt bzw. teilweise sogar überlegen ist, halten wir es dennoch für den wissenschaftlichen Einsatz geeignet.

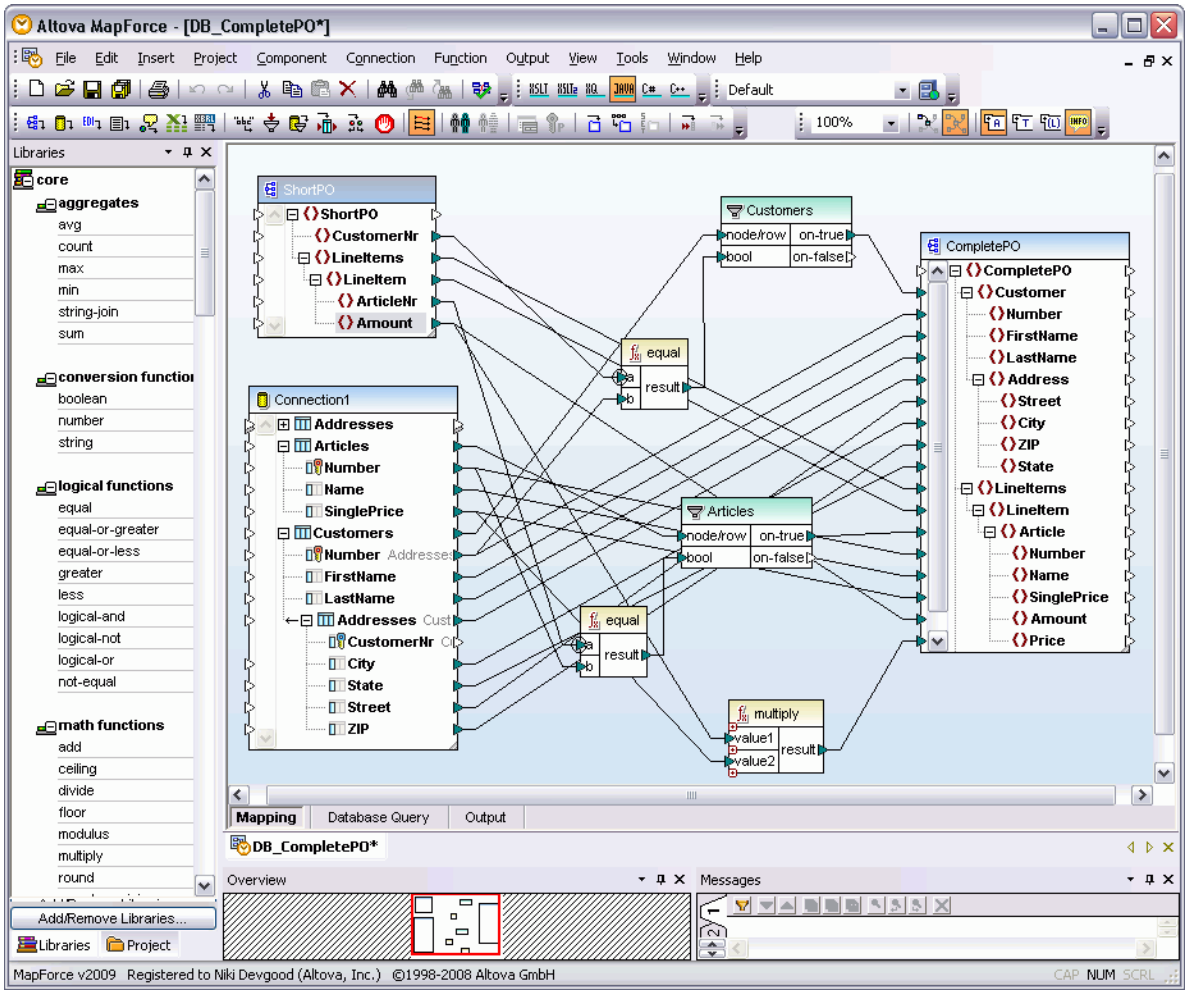

Abbildung 3.1: Benutzungsoberfläche von Altova MapForce.

Die zugrundeliegende Idee ist den Prozess der Datenintegration als Workflow zu modellieren. Dabei werden eine Menge von Quelldaten über einen Integrationsworkflow auf eine Menge von Zieldaten abgebildet. Die Erstellung dieses Workflows ist durch den Benutzer grafisch durch eine Vielzahl vordefinierter sowie durch zusätzliche benutzerdefinierte Funktionen realisierbar. Ein Screenshot der Oberfläche ist in *Abbildung 3.1* abgebildet. Der Integrationsworkflow liegt anschließend als Quellcode vor und kann damit beliebig angepasst werden. Mögliche Sprachen sind u.a. JAVA, C#, C++. Der Integrationsworkflow wird in der Regel asynchron angestoßen und endet in der Bereitstellung der integrierten Daten.

Die Ergebnisse der Evaluation von Altova MapForce nach dem Evaluationsschema finden sich in Kapitel 6.

3.8 OGSA-DAI

Open Grid Service Architecture Data Access and Integration (OGSA-DAI) [OGSa] ist eine allgemein verwendbare und erweiterbare Middleware für daten-zentrierte Workflows (siehe z.B. *Abbildung 3.2*). Die Anwendung bzw. der Benutzer setzt dabei über eine Java-API vordefinierte, sowie benutzerdefinierte *Aktivitäten* zu einem Workflow zusammen und startet dessen Ausführung per Webservice. Aktivitäten können Datentransformationen oder Zugriffe auf *Ressourcen* sein. Beim Zugriff auf Ressourcen können Aktivitäten lesend oder schreibend auf heterogen strukturierte Datenquellen wie Datenbanken, Dateisysteme oder FTP-Senken zugreifen. Vorrangig entwickelt wurde OGSA-DAI für den Einsatz innerhalb einer Grid-Infrastruktur, es kann seine Dienste jedoch auch mit nur einem OGSA-DAI Server anbieten.

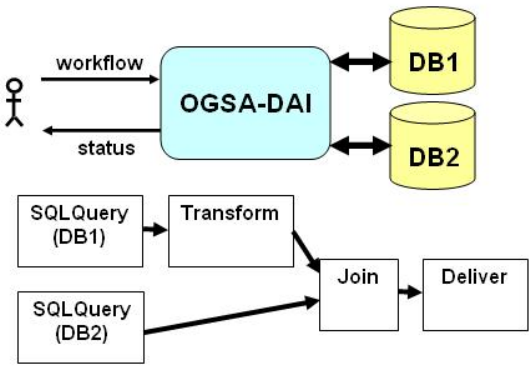

Abbildung 3.2: Möglicher Workflow zur Datenintegration, welcher mit OGSA-DAI abgearbeitet werden kann (aus [OGSa]).

Die Ergebnisse der Evaluation von OGSA-DAI nach dem Evaluationsschema finden sich in Kapitel 4.

4 Evaluation von OGSA-DAI

Dieses Kapitel beschreibt die Ergebnisse der Evaluation von OGSA-DAI [OGSa]. Allgemeine Informationen zu diesem System sowie die Definition der Begriffe *Aktivität* und *Ressource* wurden bereits im Abschnitt 3.8 gegeben.

4.1 Klassifikation

In diesem Abschnitt wird OGSA-DAI anhand der im Evaluationsschema beschriebenen Kriterien klassifiziert.

4.1.1 Enge oder lose Integration

Aus Sicht des Workflowdesigners findet eine lose Integration statt. Er muss die lokalen Schemata der Informationssysteme kennen, um diese im Zuge der Abarbeitung der Workflows zu integrieren. Eine Anwendung, die auf OGSA-DAI zugreift, sieht jedoch nur das Schema, welches nach der Integration von OGSA-DAI bereitgestellt wird, also die Datenstruktur der Ausgabedaten des Workflows. Dieses kann sich durch Integration und Transformation völlig von den Quellschemata unterscheiden. Es handelt sich aus Sicht der Anwendung also um eine enge Integration.

4.1.2 Virtuelle oder materialisierte Integration

OGSA-DAI unterstützt synchrone und asnychrone Anfragen an die Workflows. Bei synchronen Anfragen werden der Anwendung mögliche Daten nach Abarbeitung des Workflows zurückgegeben und nicht gespeichert. Dies ist also eine virtuelle Integration. Bei asynchronen Anfragen erhält die Anwendung zunächst kein Ergebnis zurück, kann aber mittels eines Webservice-Aufrufes prüfen, wie weit die Abarbeitung des Workflows vorangeschritten ist. Dabei können Daten während der Abarbeitung des Workflows an verschiedene Senken ausgeliefert werden (z.B. ftp-Upload), die dann von Anwendungen asnychron abgefragt werden können. In diesem Fall handelt es sich also um eine materialisierte Integration. Eine gleichzeitige Auslieferung der Daten an eine Senke auch bei synchronen Anfragen ist bei einem entsprechenden Workflow natürlich auch denkbar.

4.2 Evaluationsschema nach gebotener Funktionalität

Der folgende Abschnitt beschreibt die detaillierte Untersuchung von OGSA-DAI anhand seiner Funktionalitäten. Die Funktionalitäten wurden gemäß der Anordnung im Evaluationsschema analysiert, und werden auch in dieser Reihenfolge beschrieben.

4.2.1 Erweiterbarkeit und Änderbarkeit

OGSA-DAI ist sehr gut erweiterbar und änderbar. Sowohl Aktivitäten (z.B. eine eigene Datentransformation zum Auftrennen von benutzerdefiniert kodierten Tupeln) als auch OGSA-DAI-Ressourcen (z.B. Zugriff auf eine neuartige XML-Datenbank) können selbst in Java dazuprogrammiert werden, auf den Server deployt werden und so von Workflows genutzt werden.

OGSA-DAI ist ein Open-Source-System. Damit ist der Code auch frei änderbar und erweiterbar.

4.2.2 Lesender oder schreibender Zugriff auf lokale Datenquellen

Bei der Ausführung des Workflows ist es möglich, auf die unterliegenden Datenquellen (Data Resources) auch schreibend zuzugreifen. Allerdings verhält sich OGSA-DAI gegenüber dem Client nicht als klassische Datenbank, auf die schreibender und lesender Zugriff möglich wäre, sondern die Definition des Workflows lässt die Möglichkeit zu, Datensenken (Data Sink) und Datenquellen (Data Source) zu definieren. Mittels der Datensenken kann die Anwendung Datenströme auf den OGSA-DAI-Server übertragen. Wie diese Datenströme nun weiterverarbeitet werden und ob sie bis auf die unterliegenden Datenquellen durch geschrieben werden, hängt vom konkreten Workflow ab.

4.2.3 Unterstützte Zugriffsmethoden

Der Zugriff auf OGSA-DAI funktioniert über Webservice-Aufrufe, die also eine parameterisierte Funktion darstellen. Der Zugriff auf die Daten selbst, nachdem sie z.B. bei einem asynchronen Aufruf materialisiert als Resultat des Workflows in Datensenken ausgeliefert wurden, kann durch die Anwendung mittels von den Datensenken angebotener Zugriffsmethoden erfolgen, z.B. über Datenbanktreiber, also durch eine Abfragesprache.

4.2.4 Integrierbare Zugriffsquellen

Mögliche Ressourcen auf die OSGA-DAI schreibend und lesend zugreifen kann sind:

- Alle relationalen Datenbanken, für die es einen JDBC-Treiber gibt, u.a. MySql, IBM DB2, MS-SQL Server, Oracle und PostgreSQL.

- XML-Datenbanken: Davon alle, für die es einen Java-Treiber gibt.
- Dateisystem: Zugriff auf Verzeichnisstrukturen und Dateien.
- Remote-Ressourcen: Von entfernten OGSA-DAI Servern angebotene Ressourcen, z.B. bei einer Grid-Infrastruktur.
- Selbst entwickelte Ressourcen: Diese können benutzerdefiniert als Java-Programme entwickelt und zur Nutzung in Workflows bereitgestellt werden.

4.2.5 Autonomie der lokalen Datenquellen

Ausführungsautonomie: Lokale Informationssysteme besitzen die selbe Ausführungsautonomie wie im lokalen Fall. Bei synchroner als auch asnychroner Abarbeitung des Workflows wartet OGSA-DAI auf das Resultat der Anfrage des jeweiligen lokalen Informationssystems. An der möglichen Weiterverarbeitung der Daten im OGSA-DAI-Server ist das lokale Informationssystem nicht mehr beteiligt. Die lokalen Informationssysteme koordinieren oder kommunizieren nicht miteinander. Einschränkungen der Ausführungsautonomie wären auch überhaupt nicht nötig, da OGSA-DAI z.B. keine globalen Transaktionen unterstützt.

Kommunikationsautonomie: Kommunikationsautonomie gilt hier ebenso mit der selben Begründung wie bei der Ausführungsautonomie.

Entwurfsautonomie: Entwurfsautonomie ist aus Sicht der Anwendung insofern gegeben, als dass strukturelle und semantische Änderungen des lokalen Informationssystems durch eine Anpassung des Workflows aufgelöst werden können. Aus Sicht des Workflow-Designers ist deshalb nur geringe (siehe „Aufzulösende Heterogenitäten“) Entwurfsautonomie gegeben.

4.2.6 Aufzulösende Heterogenitäten

Datenmodell: Datenmodellheterogenität wird automatisch aufgelöst. Der Anwendung ist verborgen, in welchem Datenmodell die genutzten Ressourcen gespeichert sind, und der Workflow-Designer kann auf die Ergebnisse von Anfragen unterschiedlicher Datenmodelle in gleicher Weise zugreifen.

Zugriffssprache: Heterogenität der Zugriffsprache muss manuell aufgelöst werden. Der Zugriff auf eine Ressource kann im Rahmen einer OGSA-DAI-Aktivität nur mit der Ressourcen-spezifischen Anfragesprache erfolgen, in die der Workflow-Desginer die Anfrage entsprechend des vom lokalen Informationssystems verstandenen Dialekts schreiben muss.

Syntax und Semantik: Heterogenitäten der Syntax und Semantik müssen manuell aufgelöst werden. Datenintegration und -transformation muss entweder durch passende bereits existierende Aktivitäten innerhalb des Workflows realisiert werden oder sonst durch eigene Aktivitäten selbst programmiert werden. Für das Zuweisen von Spalten oder Teilen eines Schemas auf andere Teilschemata oder das Transformieren von unterschiedlich dargestellten Daten in ein gemeinsames Format ist der Workflow-Designer also selbst verantwortlich.

4.2.7 Globale Zugriffsschicht

OGSA-DAI besitzt keine globale Zugriffsschicht. Das Ergebnis eines Datenintegrationsworkflows kann zwar als Schicht angesehen werden, die den Zugriff auf die lokalen Daten abstrahiert, allerdings ist die Definition aus 2.2.7 nicht erfüllt, da es sich um einen Integrationsworkflow handelt. Damit erübrigen sich auch die erweiterten Analysen bezüglich der Sprache der globalen Zugriffsschicht, der globalen Transaktionen und deren Eigenschaften, sowie der Betrachtung von globalen Optimierungen.

4.2.8 Anbindung an Workflow-Systeme oder/und Webservices

OGSA-DAI bietet WSDL-Schnittstellen zur Übertragung, Ausführung und Statusabfrage der Workflows an. Damit können die Workflows als Webservices und damit auch von BPEL-Prozessen oder anderen Webservice-basierten Workflow-Systemen aufgerufen werden. Jedoch kann OGSA-DAI von Haus aus nicht auf fremde Webservices als Ressource zugreifen, sondern nur auf andere OGSA-DAI Knoten. Als selbst programmierte Ressource wäre eine Anbindung eines Webservices als Ressource möglich.

4.3 Sonstige Anforderungen

Im folgenden Abschnitt sind weitere Anforderungen und Qualitätsmerkmale von OGSA-DAI beschrieben. Dies umfasst insbesondere nichtfunktionale Merkmale.

4.3.1 Status des Projekts

Seit 2002 wird OGSA-DAI von einem Team an der University of Edinburgh stetig weiterentwickelt und erhält seine Mittel vom UK e-Science Programm. Die aktuelle Version 3.1 ist vom Mai 2009. Die neue Version 4.0 ist für Januar 2010 angekündigt.

4.3.2 Einsatzbereich

Laut Hersteller ist der Einsatzbereich von OGSA-DAI jegliche Datenintegration, wobei die Größe der Daten laut Hersteller „bis zum Größenmaßstab des Internets“ [OGSa] skalierbar sein soll. Demnach ist es in nahezu jedem wissenschaftlichen Bereich einsetzbar.

4.3.3 Verbreitung

Laut [OGSb] ist OGSA-DAI als Middleware-Komponente Teil von über 50 Projekten. Diese sind größtenteils im biotechnologischen oder geologischen Bereich angesiedelt, oder OGSA-DAI wird darin in der verteilten Grid-Architektur eingesetzt.

4.3.4 Kaufpreis/Lizenz

OGSA-DAI ist freie Software und wird unter den Bedingungen der Apache 2.0-Lizenz vertrieben.

4.3.5 Unterstützte Plattformen

Unterstützt werden alle Plattformen, auf denen die Abhängigkeiten Java, Apache Tomcat und Ant existieren. Dies sind u.a. Windows, Linux und Mac OS.

4.3.6 Installation und Administration

Die Installation und Administration von OGSA-DAI ist einer der größten Kritikpunkte an diesem System. Die Installation ist mühsam: Nach manueller Installation der vorausgesetzten Komponenten Apache Tomcat, Ant und des Java SDK müssen Umgebungsvariablen und Konfigurationsdateien eingerichtet bzw. geändert werden und OGSA-DAI auf den Tomcat-Server deployt werden. Anschließend müssen die verwendeten Ressourcen auf OGSA-DAI deployt werden. Hierfür muss für jede Ressource eine Konfigurationsdatei erstellt werden und diese wiederum über ant-Kommandozeilenaufrufe integriert werden. Erst danach kann diese in einem Workflow genutzt werden.

4.3.7 Benutzungsfreundlichkeit

Die Anwendung hat Webservice-Schnittstellen zur Verfügung für die Übertragung und Nutzung eines Workflows. Entsprechend gibt es keine Präsentationsschicht.

Um einen Workflow zu definieren, muss man diesen unter Nutzung der OGSA-DAI API in Java prorammieren, indem man Eingänge und Ausgänge von Aktivitäten miteinander und mit Ressourcen verbindet. Hierfür sind Programmierkenntnisse nötig. Eventuell könnte man

dies vereinfachen, indem man eine Präsentationsschicht oberhalb von OGSA-DAI entwickeln lässt.

4.3.8 Verfügbarkeit von Hilfestellungen

Die von den Entwicklern von OGSA-DAI bereitgestellte Dokumentation beschreibt die Dienste von OGSA-DAI recht ausführlich und genau. Sie ist gegliedert in Abschnitte für Deployer, Benutzer der bereits mitgelieferten Beispiel-Clients, Entwickler eigener Clients und Serverentwickler. Für Deployer wird beschrieben, wie Ressourcen und Aktivitäten definiert und auf den Server deployt werden können. Für Serverentwickler wird das Entwickeln eigener Ressourcen oder Aktivitäten beschrieben. Punktuell sind kleine Fehler oder Ungenauigkeiten vorhanden, wie z.B. für Windows unverändert übernommene Linux-Kommandozeilen-Syntax.

Die Beispiel-Clients sind mitgelieferte Java-Programme, die zeigen, wie man auf auf den OGSA-DAI-Server deployte Ressourcen wie Webservices von Java aus zugreift.

Darüberhinaus gibt es auf Sourceforge eine Online-Community mit weiteren Möglichkeiten zur gegenseitigen Hilfe.

4.3.9 Performanz

Workflows werden wenn möglich parallel auf dem OGSA-DAI Server ausgeführt und erreichen den Client so schneller und ressourcenschonender. Außerdem kann OGSA-DAI auf mehreren Rechnern innerhalb einer Grid-Infrastruktur eingesetzt werden und ermöglicht so verteiltes Rechnen.

Ansonsten konnte mangels Vergleichbarkeit mit anderen Systemen dieses Kriterium nicht quantitativ getestet werden.

5 Evaluation von BioMart

Dieses Kapitel beschreibt die Ergebnisse der Evaluation von BioMart. Allgemeine Informationen zu diesem System wurden bereits im Abschnitt 3.2 gegeben.

5.1 Klassifikation

In diesem Abschnitt wird BioMart anhand der im Evaluationsschema beschriebenen Kriterien klassifiziert.

5.1.1 Enge oder lose Integration

Für jede Datenquelle von BioMart wird eine Sicht verwendet. Da diese die Datenquelle selbst vor der Anwendung verbergen, handelt es sich aus deren Perspektive um eine enge Integration. Zur Erstellung dieser Sichten muss der Entwickler die lokalen Schemata kennen. Aus diesem Grund handelt es sich für diesen um eine lose Integration. BioMart ist konzeptuell für die Integration von Bioinformatikdatenbanken ausgelegt, bei denen integrierte Sichten keine Rolle spielen.

5.1.2 Virtuelle oder materialisierte Integration

Die Integration ist virtuell: Marts sind auf Datenbankebene als *Sicht* realisiert und es findet kein Caching statt.

5.2 Evaluationsschema nach gebotener Funktionalität

Der folgenden Abschnitt beschreibt die detaillierte Untersuchung von BioMart anhand seiner Funktionalitäten. Die Funktionalitäten wurden gemäß der Anordnung im Evaluationsschema analysiert, und werden auch in dieser Reihenfolge beschrieben.

5.2.1 Erweiterbarkeit und Änderbarkeit

Das Hinzufügen neuer Datenquellen in einen BioMart-Server ist jederzeit möglich. Das Ändern der Datenquellen ist ebenfalls einfach möglich. Da die Integrationsweise für jede Anfrage neu festgelegt wird, treten keine Konsistenzprobleme bei Ändern der Sichten auf.

BioMart an sich ist Open Source und damit auch frei änder- und erweiterbar. Es gibt beispielsweise bereits eine Implementierung der Clientbibliothek für Ruby ([Biod]), sowie eine vereinfachte Weboberfläche mit einem einzigen Suchfeld ([Mar]).

Durch eine eigene Implementierung des Serverprotokolls können BioMart prinzipiell auch andere Datenquellen als relationale Datenbanken bereitgestellt werden, wenn auch mit erheblichem Aufwand.

5.2.2 Lesender oder schreibender Zugriff auf lokale Datenquellen

BioMart erlaubt nur lesenden Zugriff auf die Datenquellen.

5.2.3 Unterstützte Zugriffsmethoden

Auf BioMart kann über die bereitgestellten grafischen Java-Programme, ein grafisches Web-Interface (*MartView*), einen Webservice (*MartService*) oder mittels einer in Perl definierten funktionsorientierten API zugegriffen werden. Alle Methoden stellen also eine parameterisierte Funktion dar.

5.2.4 Integrierbare Zugriffsquellen

BioMart kann nur auf die relationalen Datenbanken MySQL, PostgreSQL und Oracle zugreifen, nicht jedoch auf Dateien oder XML-Datenbanken. Weitere Zugriffsquellen können nur mit sehr hohem Aufwand hinzugefügt werden, siehe Abschnitt 5.2.1.

5.2.5 Autonomie der lokalen Datenquellen

Ausführungsautonomie: Die lokalen Datenquellen besitzen dieselbe Ausführungsautonomie wie im lokalen Fall. BioMart wartet nach einer Anfrage auf die Abarbeitung durch das Datenbanksystem. Das Datenintegrationssystem erscheint einer Datenquelle nur als ein weiterer Client, und es konnten keine Hinweise auf die Verwendung globaler Transaktionen gefunden werden.

Kommunikationsautonomie: Kommunikationsautonomie besteht. Das Datenintegrationssystem erscheint einer Datenquelle nur als ein weiterer Client. Es findet keine Koordination mit anderen Datenquellen statt.

Entwurfsautonomie: Datenquellen haben prinizpiell Entwurfsautonomie; lediglich bei der Gestaltung der Sichten, die von BioMart verwendet werden, müssen sie sich an den Anforderungen der Datenintegration orientieren.

5.2.6 Aufzulösende Heterogenitäten

Datenmodell: Alle Datenmodelle sind grundsätzlich relational, wobei ein BioMart-Server auch auf andere BioMart-Server zugreifen kann, die dann selbst wiederum ein relationales Modell anbieten. Dieser Punkt ist also nicht zutreffend, da keine Datenmodellheterogenität aufgelöst wird.

Zugriffssprache: BioMart kann via SQL auf die unterstützten relationalen Datenbanksysteme und via *MartService*-Webservice auf andere BioMart-Server zugreifen. Heterogenitäten werden aufgelöst.

Syntax und Semantik: Heterogenitäten von Syntax und Semantik müssen manuell aufgelöst werden, da die Verknüpfung mehrerer Datenquellen durch den Benutzer vorgenommen wird.

5.2.7 Globale Zugriffsschicht

BioMart bietet prinzipiell eine globale Zugriffsschicht an; allerdings werden keine Schema-Unterschiede zwischen Datenquellen aufgelöst, da diese im Allgemeinen zu heterogen sind. Es wurden keinerlei Hinweise auf die Verwendung globaler Transaktionen oder globaler Optimierung der Anfragen gefunden. Als Abfragesprache wird die BioMart-API in verschiedenen Varianten (siehe Kapitel 5.2.3 – Unterstützte Zugriffsmethoden) verwendet.

5.2.8 Anbindung an Workflow-Systeme oder/und Webservices

BioMart bietet WSDL-Schnittstellen für Abfragen an einen BioMart-Server, außerdem können eigene Prozesse auf die BioMart-APIs in Perl und Java zugreifen. BioMart selbst kann jedoch nur auf andere BioMart-Server und Datenbanken zugreifen, nicht auf Webservices.

5.3 Sonstige Anforderungen

Im folgenden Abschnitt sind weitere Anforderungen und Qualitätsmerkmale von BioMart beschrieben. Dies umfasst insbesondere nichtfunktionale Merkmale.

5.3.1 Status des Projekts

Das aktuelle Release 0.7 wurde Mitte 2008 veröffentlicht. BioMart wird aktiv weiterentwickelt und finanziert vom European Molecular Biology Laboratory (EMBL) und dem Ontario Institute for Cancer Research.

5.3.2 Einsatzbereich

Primäres Einsatzgebiet ist die Bioinformatik, wo der zentrale Server *BioMart Central Portal* www.biomart.org als Hub zum Zugriff auf eine Vielzahl biologischer Datenbanken dient und im Durchschnitt mehr als eine Million Anfragen pro Monat beantwortet (Stand März 2009, [Bioc]).

5.3.3 Verbreitung

BioMart wird von zahlreichen wissenschaftlichen Institutionen der Biologie und Bioinformatik eingesetzt. Die BioMart-Website gibt beispielhaft etwa 20 Institutionen an, darunter Datenbanken des *European Molecular Biology Laboratory* des *European Bioinformatics Institute*, wobei letzteres auch an der Weiterentwicklung von BioMart beteiligt ist.

5.3.4 Kaufpreis

BioMart ist frei und quelloffen unter den Bedingungen der GNU Lesser General Public License (LGPL) verfügbar.

5.3.5 Unterstützte Plattformen

Die Java-Komponenten *MartJ* benötigen Java 1.3 oder höher und werden für Mac OS X, Windows und Linux/UNIX-Systeme ausgeliefert. Die Perl-Komponenten *BioMart-Perl* benötigen Perl 5.6, und, im Falle einer Webserver-Installation, den Webserver Apache.

5.3.6 Installation und Administration

Die Installationsanleitung von *BioMart-Perl* verlangt die eigene Kompilierung mehrerer Anwendungen (u.a. Apache, mod_perl) mit speziellen Parametern auf einem Linux-System. Des Weiteren bezieht sich die Installationsanleitung aus dem Jahr 2006 auf nicht mehr verfügbare Versionen dieser Anwendungen. Im Verlauf dieses Prozesses traten auf allen zur Verfügung stehenden Linux-Systemen Probleme auf. Aus diesem Grund wurde auf die lokale Installation eines BioMart-Servers verzichtet, stattdessen wurden die Funktionen der Weboberfläche mit Hilfe des Central Portal evaluiert.

5.3.7 Benutzungsfreundlichkeit

Die Bedienung von BioMart ist, sinnvoll umgesetzte *Marts* vorausgesetzt, relativ einfach: Dann wird auf BioMart primär via der Weboberfläche *MartView* zugegriffen. Diese ermöglicht die Zusammenstellung eigener Abfragen, den Download der Ergebnisse und der Abfragen selbst. Diese können dann beispielsweise in *MartService*-Anfragen oder zum Zugriff auf die Perl-API verwendet werden können.

5.3.8 Verfügbarkeit von Hilfestellungen

BioMart bietet ausführliche Installations- und Bedienungsanleitungen, wobei erstere auf dem Stand von 2006 ist und viele darin empfohlene Downloads (z.B. Apache-Webserver in Version 2.2.3) nicht mehr verfügbar sind. Das Format von Registry-Dateien zur Definition verfügbarer Datenquellen ist nicht explizit, sondern nur über mitgelieferte Beispiele dokumentiert, was das Verständnis erheblich erschwert.

5.3.9 Performanz

Bei der Verwendung des *BioMart Central Portal* ist die Performanz befriedigend: Obwohl dieses auf verteilte Datenquellen über das Internet zugreift, werden die sehr umfangreichen Ergebnisse innerhalb weniger Sekunden angezeigt. Im Rahmen dieser Evaluation wurde nicht geprüft, ob der Zugriff auf vergleichbare Datenquellen in lokalen Netzwerken die Geschwindigkeit erhöht.

6 Evaluation von Altova MapForce

Dieses Kapitel beschreibt die Ergebnisse der Evaluation von Altova Mapforce anhand des bereits beschriebenen Evaluationsschemas. Allgemeine Informationen zu diesem System wurden bereits im Abschnitt 3.7 gegeben.

6.1 Klassifikation

In diesem Abschnitt wird Altova MapForce zunächst anhand der im Evaluationsschema beschriebenen Kriterien klassifiziert. Zusammenfassend kann Altova MapForce als ein Integrationssystem mit materialisierter und enger Integration bezeichnet werden.

6.1.1 Enge oder lose Integration

Aus Sicht des Workflowdesigners findet eine lose Integration statt. Er muss die lokalen Schemata der Informationssysteme kennen, um diese im Zuge der Abarbeitung der Workflows zu integrieren. Aus Sicht einer Anwendung verwendet Altova MapForce eine enge Integration. Das Ergebnis des Integrationsworkflows endet in der Bereitstellung der Daten in einem zuvor definierten Schema. Dieses Schema ist strukturell nicht abhängig von der Struktur der zu integrierenden Schemata der Quelldaten, sondern wird im Rahmen der Workflowdefinition festgelegt. Es kann beispielsweise eine völlig andere Datenstruktur aufweisen als die Quelldaten. Während der Integration können beispielsweise durch die Kombination vorhandener Daten neue Zusammenhänge erstellt werden, durch die Struktur des Schemas der bereitgestellten Daten weit komplexer sein kann als die Summe aller Schemata der Quelldaten.

6.1.2 Virtuelle oder materialisierte Integration

Altova MapForce kann durch den anpassbaren Quellcode sehr flexibel eingesetzt werden. Der normale Einsatz ist in der Regel die asynchrone Ausführung des Workflows und die Bereitstellung des Ergebnis in materialisierter Form. Es ist jedoch auch denkbar, den Code direkt anzusteuern, sodass er synchron aufgerufen wird. In diesem Fall kann das Ergebnis sowohl lediglich zurückgegeben oder materialisiert zur Verfügung gestellt werden. Im ersten Fall entsteht eine virtuelle Integration, im zweiten Fall entsteht eine materialisierte Integration.

6.2 Evaluationsschema nach gebotener Funktionalität

Der folgende Abschnitt beschreibt die detaillierte Untersuchung von Altova MapForce anhand seiner Funktionalitäten. Die Funktionalitäten wurden gemäß der Anordnung im Evaluationsschema analysiert, und werden auch in dieser Reihenfolge beschrieben.

6.2.1 Erweiterbarkeit und Änderbarkeit

Die durch Altova MapForce erstellbaren Workflows sind von Grund auf sehr flexibel. Die Unterstützung der Datenquellen durch die Anbindung per ODBC ist sehr umfangreich. U.a. sind dies Microsoft® SQL Server, IBM DB2, Oracle, Sybase, MySQL und PostgreSQL. Zusätzlich kann jeder Workflow beliebig erweitert werden, da das Ergebnis der grafischen Workflowerstellung als Quellcode einer bestimmten Sprache vorliegt. Unterstützte Sprachen sind XSLT, XQuery, JAVA, C# und C++. Im Quellcode kann ein Integrationsworkflow entsprechend der Mächtigkeit der Programmiersprache erweitert werden. Es ist auch denkbar eine eigene Bibliotheken zu erstellen, die gewissen Standarderweiterungen bereitstellen, und diese so mit geringem Aufwand in das System zu integrieren.

6.2.2 Lesender oder schreibender Zugriff auf lokale Datenquellen

Hier lässt sich Altova MapForce wie auch OGSA-DAI nicht eindeutig einordnen. Bei der Ausführung des Workflows ist es möglich, auf beliebige Datenquellen auch schreibend zuzugreifen. Allerdings verhält sich Altova MapForce gegenüber dem Client nicht als klassische Datenbank, auf die schreibender und lesender Zugriff möglich wäre, sondern der Client hat die Möglichkeit Daten an den Integrationsprozess zu übergeben. Wie diese Daten nun weiterverarbeitet werden und ob sie auf den unterliegenden Datenquellen geschrieben werden hängt vom konkreten Workflow ab.

Ein Beispiel wäre die Materialisierung von Integrationsergebnissen. Danach könnten diese durch die Clientanwendung verändert werden. Anschließend würde ein spezieller Update-Workflow aufgerufen, der die geänderten Daten in die zugrundeliegenden Systeme zurücküberträgt. Die Funktionsweise ist analog zu OGSA-DAI und den meisten analysierten Datenintegrationssystemen.

6.2.3 Unterstützte Zugriffsmethoden

Da die Ausführung eines Altova MapForce Integrationsworkflows nicht festgelegt ist, kann die Ausführung, je nach Aufruf des Clients, auf verschiedene Arten erfolgen. Zum einen kann es sich um einen synchronen Aufruf handeln, wenn der Client auf die Beendigung des Workflows wartet, zum anderen kann es sich um einen asynchronen Aufruf handeln, wenn die Ausführung des Workflows lediglich ausgelöst wird, jedoch der Client nicht auf die Beendigung des Workflows wartet.

Im asnychronen Fall kann der Zugriff auf die materialisierten Ergebnisse der Integration in der Regel als Abfrage ausgeführt werden. Je nach Datenbank oder Dateiformat ist auch ein Zugriff über parameterisierte Funktionen oder Browsing möglich.

Im synchronen Fall, in dem das Ergebnis direkt zurückgegeben wird, kann der Zugriff als parameterisierte Funktion angesehen werden (virtuelle Integration).

6.2.4 Integrierbare Zugriffsquellen

Mögliche Zugriffsquellen, auf die Altova MapForce schreibend und lesend zugreifen kann sind:

- Alle relationalen Datenbanken, für die es einen ODBC-Treiber gibt, u.a. MySql, IBM DB2, MS-SQL Server, Oracle und PostgreSQL.
- XML-Dateien.
- Webservices
- Dateisysteme (Verzeichnisstrukturen und Dateizugriff)

6.2.5 Autonomie der lokalen Datenquellen

Altova MapForce lässt den einzelnen lokalen Datenquellen weitgehende Autonomie, da durch den Integrationsworkflow lediglich eine Art integrative Sicht über die einzelnen lokalen Datenquellen bereitgestellt wird. Nachfolgend werden die einzelnen Arten der Autonomie aus dem Evaluationsschema betrachtet:

Ausführungsautonomie: Lokale Datenquellen sind bezüglich der Ausführung vollkommen autonom. Die einzelnen Schritte des Integrationsworkflows werden sequenziell oder parallel verarbeitet. Sie beeinflussen sich und auch andere Zugriffe auf die Datenquellen gegenseitig nicht, und es findet keine Koordination statt, zum Beispiel bezüglich der ACID-Transaktionseigenschaften.

Kommunikationsautonomie: Analog zur Ausführungsautonomie sind die lokalen Datenquellen auch bezüglich der Kommunikation vollkommen autonom.

Entwurfsautonomie: Die lokalen Datenquellen sind bezüglich des Entwurfs nicht vollständig autonom. Eine Entwurfsänderung führt in der Regel dazu, dass der Workflow der Integration angepasst werden muss. Bezüglich der Semantik ist ebenfalls keine Autonomie der lokalen Datenquellen vorhanden. Eine semantische Änderung muss ebenfalls über den Workflow der Integration so angepasst werden, dass die Semantik der Gesamtdaten korrekt bleibt.

6.2.6 Aufzulösende Heterogenitäten

Datenmodell: Datenmodellheterogenität wird automatisch aufgelöst. Der Anwendung ist verborgen, in welchem Datenmodell die genutzten Ressourcen gespeichert sind, und der Workflow-Designer kann auf die Ergebnisse von Anfragen unterschiedlicher Datenmodelle in gleicher Weise zugreifen.

Zugriffssprache: Manuell. Der Zugriff auf ein lokales Informationssystem kann im Rahmen eines Workflow-Elements nur durch die zum lokalen Informationssystem passende Zugriffssprache erfolgen, in die der Workflow-Designer die Anfrage entsprechend des vom lokalen Informationssystems verstandenen Dialekts schreiben muss

Syntax und Semantik: Manuell. Datenintegration und -transformation muss entweder durch passende, bereits existierende Funktionen realisiert werden oder es müssen sonst eigene Funktionen programmiert werden. Es ist z.B. bereits eine spezielle Funktion „Zuordnungstabelle“ vorhanden, mit der Konflikte wegen Synonymen und Homonymen aufgelöst werden können.

6.2.7 Globale Zugriffsschicht

Altova MapForce besitzt keine echte globale Zugriffsschicht. Es kann das Ergebnis eines Datenintegrationsworkflow als Schicht angesehen werden, die generell den Zugriff auf die lokalen Daten abstrahiert, allerdings ist die Definition aus 2.2.7 nicht erfüllt, da es sich um einen Integrationsworkflow handelt. Damit erübrigen sich auch die erweiterten Analysen bzgl. der Sprache der globalen Zugriffsschicht, der globalen Transaktionen und deren Eigenschaften, sowie der Betrachtung von globalen Optimierungen.

6.2.8 Anbindung an Workflow-Systeme oder/und Webservices

Altova MapForce kann auf Webservices zugreifen sowohl um Daten abzufragen als auch um Daten über Parameter an Webservices zu liefern. Der resultierende Altova MapForce-Code kann leicht als Webservice veröffentlicht werden. Damit können die Workflows als Webservices und damit auch von BPEL-Prozessen oder anderen Webservice-basierten Workflow-Systemen aufgerufen werden.

6.3 Sonstige Anforderungen

Im folgenden Abschnitt sind weitere Anforderungen und Qualitätsmerkmale von Altova MapForce beschrieben. Dies umfasst insbesondere nichtfunktionale Merkmale.

6.3.1 Status des Projekts

Das Produkt wird kommerziell durch die Firma Altova entwickelt und vertrieben. Die aktuelle Version ist die 2009. Diese ist in drei Editionen (Standard, Professional und Enterprise) mit jeweils verschiedenem Funktionsumfang verfügbar. Für die Evaluation in diesem Dokument wurde die Enterprise Edition betrachtet.

In der Professional Edition fehlt im Wesentlichen die Unterstützung für den Zugriff auf Webservices, sowie für einige spezielle Datenformate wie EDIFACT [Alta] für den einheitlichen Austausch im Geschäftsverkehr. Die Standard Edition unterstützt lediglich den Zugriff auf XML-Dateien. Ein ausführlicher Editionsvergleich ist auf der Altova-Webseite [Alta] verfügbar.

6.3.2 Einsatzbereich

Da auf den Internetseiten von Altova hierzu keine Informationen verfügbar waren wurde zu den von Altova gedachten Einsatzbereichen und für Referenzprojekte versucht mit Altova Kontakt aufzunehmen. Der Versuch einer Kontaktaufnahme wurde jedoch von der Marketingabteilung nie beantwortet.

6.3.3 Verbreitung

Über die Verbreitung und den Erfolg am Markt standen leider auf den Internetseiten von Altova keine Informationen zur Verfügung und es wurden uns von Altova leider keine Informationen zur Verfügung gestellt. Unsere Kontaktaufnahme blieb ohne Antwort.

6.3.4 Kaufpreis

Der Kaufpreis variiert je nach Funktionsumfang zwischen 199 und 799 Euro pro Lizenz. Die Standard Edition kostet 199 Euro, die Professional Edition kostet 399 Euro und die Enterprise Edition kostet 799 Euro. Darüber hinaus gibt es diverse Service- und Support-Verträge und gewisse Staffelpreise für Mehrbenutzerbetrieb. Weitere Einzelheiten finden Sie in [Altb].

6.3.5 Unterstützte Plattformen

Altova MapForce unterstützt nur die Windows-Plattform. Auf der Webseite wird zwar Apple durch Parallels-Virtualisierung genannt, man kann dies jedoch nicht als volle Unterstützung bezeichnen.

6.3.6 Installation und Administration

Die Installation verläuft als gewöhnliche Windows-Installation sehr einfach. Das Produkt steht damit sofort zur Verfügung.

6.3.7 Benutzungsfreundlichkeit

Altova MapForce ist im Bereich Benutzungsfreundlichkeit positiv im Vergleich zu den anderen Systemen aufgefallen. Die Oberfläche ist aufgeräumt und relativ leicht verständlich. Natürlich bleibt das Thema Datenintegration eine komplexe Aufgabe und entsprechend sind manche Dialoge für den Laien nicht sofort intuitiv bedienbar.

Es existieren Plugins für Visual Studio und Eclipse. Diese Plugins ermöglichen einen besseren Entwurf der Integrationsworkflows (Siehe *Abbildung 6.1*)

6.3.8 Verfügbarkeit von Hilfestellungen

Neben einer ausführlichen, englischen Online-Dokumentation existieren im Internet eine FAQ-Sammlung sowie ein Diskussionsforum direkt vom Hersteller. Außerdem kann der Hersteller für Support-Anfragen oder für Online-Trainings kontaktiert werden.

Für weitergehende Supportanfragen steht ein Subscription-Vertrag zur Verfügung. Unabhängig davon stehen Bücher sowie eine Vielzahl Schulungen von Altova und anderen Unternehmen kostenpflichtig zur Verfügung.

6.3.9 Performanz

Die Performanz erscheint in den während der Evaluation durchgeführten Integrationen gut. Dieses Kriterium konnte jedoch mangels Vergleichsdaten nicht ausreichend im Vergleich zu anderen Systemen getestet werden.

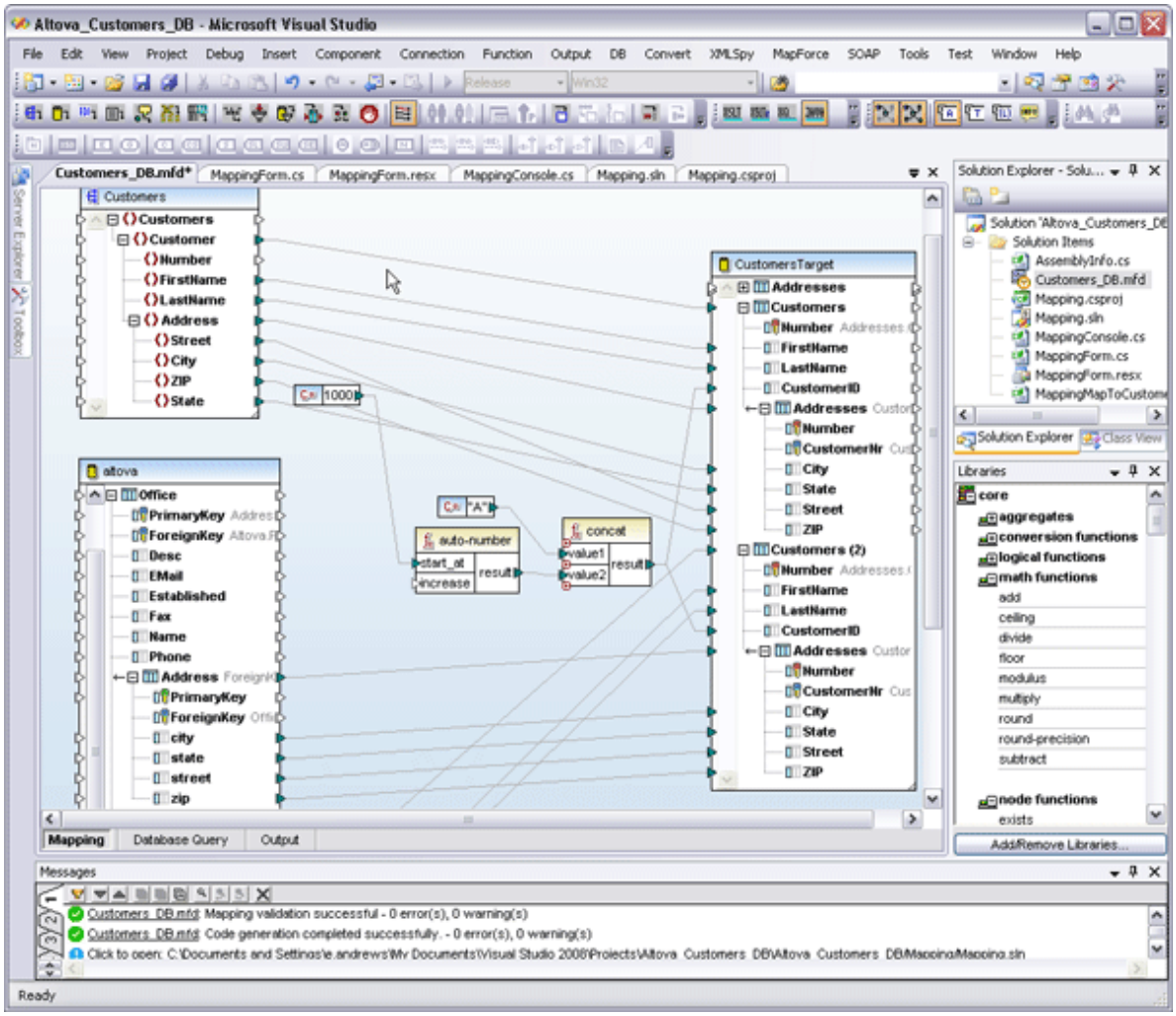

Abbildung 6.1: Altova MapForce Visual Studio Plugin.

7 Evaluation von DataDirect Data Integration Suite

In diesem Kapitel werden die Evaluationsergebnisse der *DataDirect Data Integration Suite* (DI Suite) vorgestellt.

Die Evaluation wurde mit der Evaluierungsversion der DI Suite durchgeführt, die nur sechs Tage lang lauffähig ist. Anders als erwartet wurde den Autoren keine Evaluationslizenz zugesandt, die den Zeitraum auf mindestens 14 Tage hätte verlängern können. Durch diese erhebliche Einschränkung war eine tiefergehende Evaluation nicht möglich.

7.1 Klassifikation

In diesem Abschnitt wird die Data Integration Suite anhand der im Evaluationsschema beschriebenen Kriterien klassifiziert.

7.1.1 Enge oder lose Integration

Die in der DI Suite realisierte Integration ist eng: Mittels *XQuery*-Anweisungen werden die Datenquellen in das über *XSD* oder *DTD* definierte globale Schema überführt.

7.1.2 Virtuelle oder materialisierte Integration

Die DI Suite bietet eine materialisierte Integration durch die Anforderung, die Quelldokumente immer zunächst in das Zielschema zu überführen, bevor die integrierten Daten weiterverarbeitet werden können.

7.2 Evaluationsschema nach gebotener Funktionalität

Der folgende Abschnitt beschreibt die detaillierte Untersuchung der DI Suite anhand ihrer Funktionalitäten. Die Funktionalitäten wurden gemäß der Anordnung im Evaluationsschema analysiert, und werden auch in dieser Reihenfolge beschrieben.

7.2.1 Erweiterbarkeit und Änderbarkeit

Die DI Suite ist gut erweiterbar. Die XQuery-Syntax für Integrationen erlaubt die einfache Hinzunahme weiterer Datenquellen in eine bestehende Integration.

Datenquellen können prinzipiell unabhängig voneinander geändert werden. XML-Webservices und Datenbanken werden direkt über XQuery angesprochen. Änderungen an diesen Datenquellen müssen in den der Integration zugrundeliegenden XQuery-Anweisungen berücksichtigt werden.

Sonstige Textformate können mit einem eigenen XML-Converter in ein beliebiges XML-Format konvertiert werden, bevor dieses dann ebenfalls via XQuery integriert wird. Durch diese Indirektion sind diese Ausgangsformate weitgehend unabhängig vom Zielformat, da Änderungen an den Textformaten ohne Modifikation der XQuery-Integration im XML-Converter ausgeglichen werden können.

Das Produkt selbst ist, auf Grund seiner kommerziellen Natur und da der Quellcode nicht offen verfügbar ist, nicht durch Anwender erweiterbar.

7.2.2 Lesender oder schreibender Zugriff auf lokale Datenquellen

Die DI Suite erlaubt lediglich lesenden Zugriff auf die lokalen Datenquellen.

7.2.3 Unterstützte Zugriffsmethoden

Die DI Suite kann entweder über die grafische Oberfläche Stylus Studio bedient werden oder als Java- und .NET-Komponente in Anwendungen integriert werden, um die materialisierte Datenintegration aus anderen Programmen heraus anzustoßen. Das Ergebnis der Datenintegration ist immer ein XML-Dokument, das beispielsweise über XPath angesprochen werden kann. Die DI Suite bietet keine eigene Schnittstelle zum Zugriff auf integrierte Daten.

7.2.4 Integrierbare Zugriffsquellen

Die DI Suite unterstützt direkt den Zugriff auf XML-Webservices, XML-Dokumente und relationale Datenbanken: Microsoft SQL Server, MySQL, Oracle, DB2, PostgreSQL, Sybase ASE, Informix und andere werden unterstützt. Auf eine Vielzahl von Textdokumentformaten wie CSV, EDI-Nachrichten oder Microsoft Office OpenXML kann über die XML-Converter zugegriffen werden, die diese Dokumente in ein XML-Format zur direkten Verarbeitung überführen können.

Bei der Evaluation der Integration von Datenbanken traten jedoch Probleme auf: Diese konnte auf beiden getesteten Datenbanken (MySQL Community Edition und Microsoft SQL Server 2005) aufgrund von Problemen mit den bei der DI Suite mitgelieferten Treibern nicht

erfolgreich vollzogen werden. Der eingeschränkte Evaluationszeitraum ließ keine weiteren Datenbank-Tests zu.

7.2.5 Autonomie der lokalen Datenquellen

Ausführungsautonomie: Lokale Datenquellen sind bezüglich der Ausführung vollkommen autonom. Die einzelnen Schritte der Integration werden sequenziell oder parallel verarbeitet. Sie beeinflussen sich und auch andere Zugriffe auf die Datenquellen gegenseitig nicht, und es findet keine Koordination statt, zum Beispiel bezüglich der ACID-Transaktionseigenschaften.

Kommunikationsautonomie: Analog zur Ausführungsautonomie sind die lokalen Datenquellen auch bezüglich der Kommunikation autonom.

Entwurfsautonomie: Die lokalen Datenquellen sind bezüglich des Entwurfs nicht vollständig autonom. Eine Entwurfsänderung führt in der Regel dazu, dass die XQuery-Anweisungen oder XML-Converter angepasst werden müssen. Bezüglich der Semantik ist ebenfalls keine Autonomie der lokalen Datenquellen vorhanden. Eine semantische Änderung muss ebenfalls über XQuery oder XML-Converter ausgeglichen werden, so dass die Semantik der Gesamtdaten korrekt bleibt.

7.2.6 Aufzulösende Heterogenitäten

Datenmodell: Aus Sicht der Anwendung ist die Auflösung von Datenmodellheterogenitäten automatisch, da bei der Konfiguration der Datenintegration die entsprechenden Parameter gesetzt wurden. Für den Entwickler der XQuery-Datenintegration werde jedoch keine Datenmodellheterogenitäten aufgelöst.

Zugriffssprache: Die Heterogenität der Zugriffssprache wird automatisch aufgelöst, SQL-Datenquellen, Webservices und XML-Dateien erscheinen dem Benutzer identisch. Textdateien erscheinen nach der entsprechenden Konvertierung nach XML ebenfalls hinsichtlich der Zugriffssprache homogen. Lediglich der Entwickler des entsprechenden XML-Converters muss das Ausgangsformat dieser Textdokumente kennen.

Syntax und Semantik: Heterogenitäten von Syntax und Semantik werden nicht von der DI Suite aufgelöst. Eine Anpassung der XQuery-Integration oder XML-Converter ist notwendig.

7.2.7 Globale Zugriffsschicht

Die DI Suite besitzt keine globale Zugriffsschicht: Die Datenquellen werden zunächst abgerufen, um danach in einem zweiten Schritt integriert zu werden. Hierbei handelt es sich jedoch nicht um eine Software-Komponente, wie in 2.2.7 vorausgesetzt, sondern um XML-Dokumente. Damit erübrigen sich auch die erweiterten Analysen bzgl. der Sprache der

globalen Zugriffsschicht, der globalen Transaktionen und deren Eigenschaften, sowie der Betrachtung von globalen Optimierungen.

7.2.8 Anbindung an Workflow-Systeme oder/und Webservices

Die DI Suite umfasst das Produkt *DataDirect XQuery XQueryWebService Framework* des Herstellers, mit Hilfe dessen XQuery-Anweisungen als Webservice angeboten werden können ([Datc]).

Außerdem kann die Datenintegration in einem Java- oder .NET-Programm durchgeführt werden. Integration eines solchen Programms in einen Workflow würde also diese Anbindung ebenfalls ermöglichen.

Webservices können direkt aus der DI Suite heraus aufgerufen werden.

7.3 Sonstige Anforderungen

Im folgenden Abschnitt sind weitere Anforderungen und Qualitätsmerkmale der DI Suite beschrieben. Dies umfasst insbesondere nichtfunktionale Merkmale.

7.3.1 Status des Projekts

Die aktuelle Version der DI Suite (mit *DataDirect XQuery* 4.0, *XML Converters* 4.0 und Stylus Studio 2009) wurde im Januar 2009 veröffentlicht. Die Software wird kommerziell vertrieben und unterstützt.

7.3.2 Einsatzbereich

Der Hersteller *DataDirect Technologies* ist spezialisiert auf die Entwicklung von Unternehmensanwendungen und sieht den Einsatzbereich der DI Suite in einem solchen Umfeld. So hebt das Unternehmen die Unterstützung für EDI-Nachrichtenformate in der DI Suite hervor, ein elektronischer Dokumentenstandard zum Datenaustausch zwischen Unternehmen. Auf der Website der Software ([Data]) werden als Einsatzbereich verschiedenste Industriezweige (u.a. *Automotive*, *Versicherungen* und *Software*) hervorgehoben.

7.3.3 Verbreitung

Der Hersteller nennt auf seiner Website beispielhaft etwa zehn Unternehmen unterschiedlicher Branchen, die die DI Suite erfolgreich einsetzen ([Data]). Konkrete Zahlen zur Verbreitung konnten nicht gefunden werden.

7.3.4 Kaufpreis

Der Hersteller verweist auf seiner Website in Lizenz- und Kauffragen an seine Vertriebsabteilung. Auf Nachfrage teilte *DataDirect* mit, dass die Kosten von einigen Faktoren abhängen (Prozessoranzahl und die Maximalanzahl gleichzeitig mit einem Server verbundener Benutzer werden auf der Website genannt) und bei etwa 20.000 Euro beginnen.

7.3.5 Unterstützte Plattformen

XML Converters und Stylus Studio erfordern Microsoft Windows XP oder eine neuere Windows-Version. Die prinzipiell auch unter Linux und UNIX lauffähige XQuery-Komponente liegt in der DI Suite nur als Teil einer Windows-Installationsdatei vor, so dass diese im Paket ebenfalls nur unter Windows installiert werden kann.

7.3.6 Installation und Administration

Die DI Suite wird als Installationsprogramm ausgeliefert und ist dadurch sehr einfach zu installieren. Eine weitergehende Administration ist mit dieser Software nicht nötig.

7.3.7 Benutzungsfreundlichkeit

Die Bedienung von Stylus Studio, dem Editor für XML-Converter und XQuery-Programme ist einfach: Nach kurzer Einarbeitung kann man mehrere Dokumente in ein gemeinsames Schema integrieren. Datei-Datenquellen können in das Programm eingebunden werden, Datenbankverbindungen kann man dauerhaft hinterlegen. Bei der XQuery-Erstellung kann man Elemente der lokalen Datenquellen und des Zielschemas grafisch verknüpfen. Operatoren, beispielsweise für Vergleiche, werden ähnlich elektronischer Bausteine angezeigt (siehe Abbildung 7.1), deren Ein- und Ausgänge mit Elementen aus Quell- und Zieldokument verknüpft werden. Insgesamt ist die Bedienung komfortabel.

7.3.8 Verfügbarkeit von Hilfestellungen

Der Hersteller bietet eine ausführliche Dokumentation zu seinen Produkten. Dies umfasst sowohl Installations- und Benutzungshandbücher und Quellcode-Beispiele, als auch Videos, in denen der Bildschirm eines Benutzers während der Bedienung der Software abgefilmt wurde (sog. *Screencasts*). Des Weiteren bietet der Hersteller Produktunterstützung via Ticketsystem, Telefon-Support und Webformular.

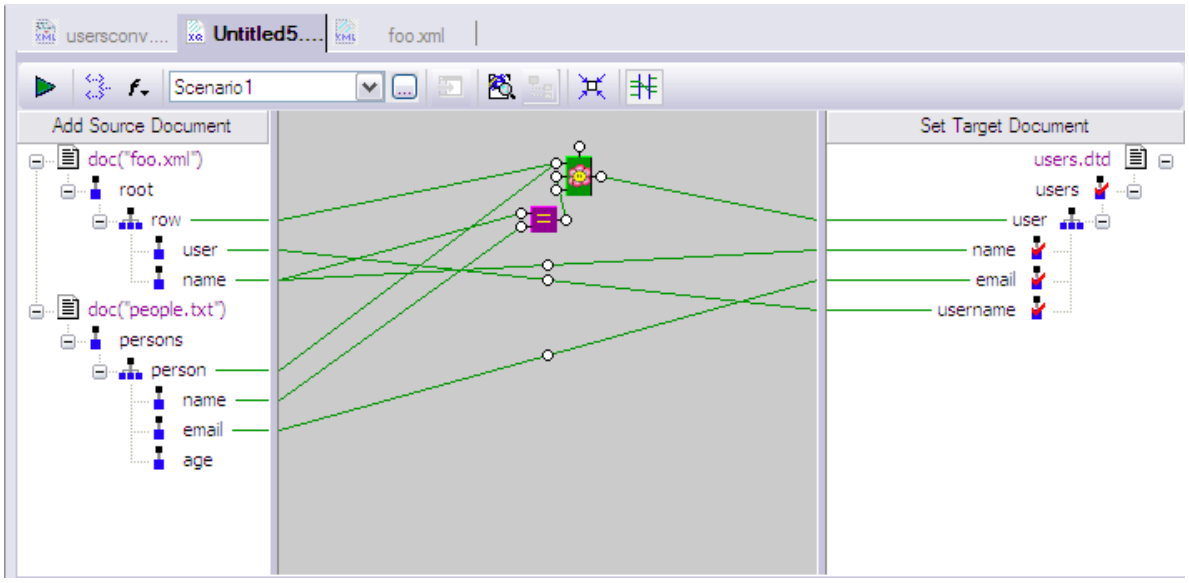

Abbildung 7.1: Ausschnitt der Benutzungsoberfläche von DataDirect Data Integration Suite.

7.3.9 Performanz

Aufgrund des sehr begrenzten Evaluationszeitraums und der Probleme bei der Anbindung an Datenbanken kann hierzu keine Aussage getroffen werden.

8 Zusammenfassung und Bewertung

Im abschließenden Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst. Dazu gehört eine Betrachtung der Unterschiede der im vorigen Abschnitt näher betrachteten Systeme, sowie eine Bewertung bzgl. einer möglichen Verwendung in der Abteilung Anwendersoftware.

8.1 Unterschiede zwischen den näher betrachteten Systemen

In diesem Abschnitt werden die zentralen Unterschiede und Besonderheiten der näher betrachteten Systeme zusammengefasst.

OGSA-DAI und MapForce bieten im Gegensatz zu den anderen Systemen eine Workflowumgebung zur Datenintegration an. Beide stellen dazu ein Framework zum Zugriff auf heterogene Datenquellen und Aktivitäten zur Datentransformation und -integration zur Verfügung.

Bei OGSA-DAI kann über die Webservice-Schnittstelle ein Workflow in Form des Java-Quellcodes übertragen werden und die Ausführung des Workflows dann von der Anwendung angestoßen werden. In MapForce kann der Workflow grafisch erstellt werden, und daraus wird ein ähnlicher Client-Code wie bei OGSA-DAI generiert. Bei Änderungen muss dieser Code neu generiert werden, oder die Änderungen müssen direkt im Code durchgeführt werden, wodurch jedoch evtl. Inkonsistenzen zwischen generiertem Code und der grafischen Workflow-Definition entstehen.

Der Hauptnachteil von BioMart im Vergleich zu den anderen Systemen liegt darin, dass BioMart nur auf relationale Datenquellen zugreifen kann, die durch Joins oder andere Operatoren der Relationenalgebra integriert werden. Die Anfragen an BioMart können via Webinterface, durch Webservices oder per APIs abgesendet werden.

Bei der Data Integration Suite werden zunächst die zu integrierenden Daten in ein XML-Format transformiert. Die Integration wird anschließend via XQuery-Anfragen durch den Benutzer mit Hilfe eines grafischen Editors definiert und durch die Anwendung aufgerufen.

8.2 Verwendung in der Abteilung Anwendersoftware

Im Projekt „Data provisioning for scientific workflows“ innerhalb des Exzellenzclusters „SimTech“ sollen Entwickler von Simulationsworkflows Datentransformationen möglichst einfach und abstrakt definieren können. Als Middleware für das dabei entstehende System könnten bestehende Datenintegrationssysteme eingesetzt werden.

Datenquellen sind entweder Ausgaben von naturwissenschaftlichen oder ingenieurwissenschaftlichen Simulationsprozessen oder Zwischenergebnisse bisheriger Integrationen. Daher ergeben sich für das einzusetzende Datenintegrationssystem die im folgenden Abschnitt beschriebenen Anforderungen.

8.2.1 Anforderungen

Verarbeitung von sehr großen Datenmengen: Alle näher betrachteten Systeme erfüllen diese Anforderung und haben generell keine Einschränkung der Größe der Daten. Bei größeren Datenmengen ist mit einer längeren Integrationszeit zu rechnen, dies konnte jedoch nicht quantifiziert werden. Dieses Kriterium wird also von allen Systemen erfüllt.

Unterstützung hochgradig heterogener Daten: Insbesondere müssen dabei folgende Anforderungen unterstützt werden:

Zugriff auf relationale Datenbanken: Wegen der Existenz von öffentlich zugänglichen relationalen Datenbanken (z.B. Swiss-Prot [Swi]) im Anwendungsgebiet ist eine Zugriffsmöglichkeit hierauf eine wichtige Anforderung. Dabei sollten möglichst alle relationalen Datenbanksysteme unterstützt werden.

Alle näher betrachteten Systeme unterstützen relationale Datenbanken als Datenquellen. Je nach System werden unterschiedliche Datenbanksysteme unterstützt, wie in den Kapiteln zur Evaluation beschrieben wurde, wobei die Unterstützung meist von der Existenz eines Datenbanktreibers anhängt, z.B. von JDBC bei OGSA-DAI und ODBC bei MapForce. Biomart kann auf MySQL, PostgreSQL und Oracle zugreifen und die DI Suite u.a. auf Microsoft SQL Server, MySQL, Oracle und IBM DB2.

Unterstützung verschiedener, teils proprietärer Dateiformate: Die Wissenschaftler im Anwendungsgebiet definieren meist ihre eigenen proprietären Datenformate und schreiben die Daten vorwiegend in einzelne Dateien im Dateisystem.

Alle näher betrachteten Systeme außer BioMart unterstützen Zugriff auf Dateien. BioMart erfüllt durch die Einschränkung auf relationale Daten dieses Kriterium nicht. Die Unterstützung der Data Integration Suite für Textformate ist besonders ausgeprägt durch die einfache Erzeugung eigener Konvertierungskomponenten nach XML.

Sensornetze: Oft müssen Daten aus Sensornetzen gelesen werden, die über ein Gateway erreichbar sind. Letzteres stellt oft eine SQL-ähnliche Anfragesprache bereit. MapForce und OGSA-DAI unterstützen dies durch benutzerdefinierte Quellen. Für die Data Integration Suite ist lediglich eine Konvertierung nach XML notwendig. BioMart könnte dieses Kriterium nur dadurch erfüllen, indem mit hohem Aufwand ein eigener BioMart-Server als Wrapper für die Quelle geschrieben wird oder der Gateway sich als eines der unterstützten relationalen Datenbanksystemen verhält.

Verarbeitung von XML-Daten: Mit Ausnahme von BioMart unterstützen alle näher betrachteten Systeme den Zugriff auf XML-Daten. BioMart erfüllt durch die Einschränkung auf relationale Daten dieses Kriterium nicht.

Verwendung innerhalb von BPEL-Prozessen: OGSA-DAI und BioMart stellen direkt Webservice-Schnittstellen zur Verfügung. Bei OGSA-DAI erlauben diese die Übertragung und Ausführung der Workflows. Bei BioMart erlaubt der MartService-Webservice das Absenden von Anfragen an den BioMart-Server. Für MapForce und die Data Integration Suite muss der generierte Code jeweils mit geringem Aufwand als Webservice veröffentlicht werden.

Zugriff auf Webservices: BioMart kann lediglich auf andere BioMart-Server per Webservice zugreifen, nicht aber auf andere Datenquellen, die ihre Daten via Webservices zur Verfügung stellen. In OGSA-DAI muss der Zugriff durch eine eigene Aktivität realisiert werden, was jedoch mit vertretbarem Aufwand möglich ist. Data Integration Suite und MapForce können nativ auf Webservices zugreifen.

Einfache Bedienung: Da die Anwender des Systems keine Informatiker sein werden, sollte die Bedienung vergleichbar zu einfachen Anwendungsprogrammen sein.

Alle Systeme außer OGSA-DAI lassen sich relativ komfortabel bedienen und stellen grundsätzliche Hilfestellungen zur Verfügung. OGSA-DAI ist wegen der Kommandozeilenlastigkeit und dem Fehlen einer eigenen Präsentationsschicht schwerer zu bedienen.

Keine Voraussetzung von Programmierkenntnissen: Da die Anwender des Systems keine Informatiker sein werden, sollen Programmierkenntnisse nicht vorausgesetzt werden.

Für OGSA-DAI, MapForce und die Data Integration Suite sind Programmierkenntnisse erforderlich. Die Workflows lassen sich in MapForce und der Data Integration Suite leicht definieren, allerdings sind für die Integration in eine Anwendung Programmierkenntnisse erforderlich. Bei OGSA-DAI muss der Workflow in Java-Code unter Nutzung der OGSA-DAI-API erstellt werden wofür Programmierkenntnisse nötig sind. BioMart kann ohne Programmierkenntnisse über die Webseite bedient werden.

8.2.2 Empfehlungen

BioMart ist für den hier beschriebenen Einsatzbereich nicht zu empfehlen. Es unterstützt nur relationale Daten und keine proprietären Dateiformate. Außerdem unterstützt BioMart nicht den Zugriff auf XML-Daten. Auf Sensornetzwerke und Webservices ist ein Zugriff nur mit sehr großem Aufwand möglich.

OGSA-DAI erfüllt zwar alle funktionalen Anforderungen bis auf den nativen Zugriff auf Webservices und ist sehr gut erweiterbar und änderbar, ist jedoch Aufgund seiner umfassende System- und Programmierkenntnisse erfordernden Bedienung für den effizienten Einsatz durch Wissenschaftler nicht zu empfehlen. Durch das Erstellen einer eigenen Präsentationsschicht könnte ein Einsatz von OGSA-DAI jedoch trotzdem interessant sein.

MapForce und Data Integration Suite erfüllen alle funktionalen Anforderungen, sind jedoch nicht vom Hersteller explizit für den wissenschaftlichen Einsatz konzipiert worden. Diese beiden Systeme bieten die Möglichkeit, grafisch generierte Abfragen als Quellcode zu exportieren. Für die Integration dieses generierten Programmcodes in bestehende Systeme sind Programmierkenntnisse erforderlich. Altova MapForce ist bereits für unter 1.000 Euro zu haben. Im Gegensatz dazu kostet die Data Integration Suite mindestens etwa 20.000 Euro, was einem umfassenden Einsatz im Rahmen des Projekts im Weg stehen könnte.

Literaturverzeichnis

[Alta] *Altova Editionsvergleich* – http://www.altova.com/de/mapforce/editionsvergleich.html (Zitiert auf Seite 41)

[Altb] *Altova Shop* – https://shop.altova.com/category.asp?category_name=MAPFORCE (Zitiert auf Seite 41)

[Aut] *AutoMed* – http://www.doc.ic.ac.uk/automed/ (Zitiert auf Seite 19)

[Bioa] *BioJava* – http://biojava.org (Zitiert auf Seite 20)

[Biob] *BioMart* – http://www.biomart.org (Zitiert auf Seite 19)

[Bioc] *BioMart Projekte* – http://nar.oxfordjournals.org/cgi/content/full/gkp265/DC1 (Zitiert auf Seite 34)

[Biod] *BioMart Ruby* – http://github.com/dazoakley/biomart/tree/master (Zitiert auf Seite 32)

[Bioe] *BioMediator* – http://www.biomediator.org (Zitiert auf Seite 20)

[Biof] *BioPerl* – http://bioperl.org (Zitiert auf Seite 20)

[Biog] *BioPython* – http://biopython.org (Zitiert auf Seite 20)

[Bioh] *BioRuby* – http://bioruby.org (Zitiert auf Seite 20)

[Bioi] *BioSQL* – http://www.biosql.org (Zitiert auf Seite 20)

[BKLW99] Susanne Busse, Ralf-Detlef Kutsche, Ulf Leser und Herbert Weber. *Federated Information Systems: Concepts, Terminology and Architectures.* Forschungsberichte des Fachbereichs Informatik, Bericht Nr. 99-9, Technische Universität Berlin, April 1999. – (Zitiert auf den Seiten 11 und 15)

[Data] *Data Integration Suite Industriezweige* – http://www.datadirect.com/products/data-integration/industry-success/all-industries/index.ssp (Zitiert auf Seite 48)

[Datb] *DataDirect Data Integration Suite* – http://www.xquery.com/data-integration (Zitiert auf Seite 21)

[Datc] *DataDirect XQuery Web Services* – http://www.xquery.com/web_services (Zitiert auf Seite 48)

[Emb] *Emboss* – http://emboss.sourceforge.net/what (Zitiert auf Seite 21)

[JTMP04] Edgar Jasper, Nerissa Tong, Peter McBrien und Alexandra Poulovassilis. *Generating and Optimising Views from Both as View Data Integration Rules*. School of Computer Science and Information Systems, Birkbeck College, University of London, 2004. (Zitiert auf Seite 19)

[KE97] A. Kemper, A. Eickler. *Datenbanksysteme - Eine Einführung*. R. Oldenbourg Verlag München Wien, 1997 (Zitiert auf Seite 13)

[Map] *MapForce* – http://www.altova.com/mapforce.html (Zitiert auf Seite 21)

[Mar] *MartSearch* – http://github.com/dazoakley/martsearch/tree/master (Zitiert auf Seite 32)

[OGSa] *OGSA-DAI* – http://www.ogsadai.org.uk (Zitiert auf den Seiten 7, 23, 25 und 29)

[OGSb] *OGSA-DAI Projekte* – http://www.ogsadai.org.uk/about/projects.php (Zitiert auf Seite 29)

[Rah94] Erhard Rahm. *Mehrrechner-Datenbanksysteme: Grundlagen der verteilten und parallelen Datenverarbeitung*. Addison-Wesley, Bonn, 1994 (Zitiert auf Seite 11)

[Rei08] Peter Reimann. *Optimization of BPEL/SQL Flows in Federated Database Systems*. Diplomarbeit am Institut für Parallele und Verteilte Systeme, Universität Stuttgart, 2008 (Zitiert auf Seite 11)

[Swi] *Swiss-Prot* – http://www.expasy.org/sprot (Zitiert auf Seite 52)

Alle URLs wurden zuletzt am 28.09.2009 geprüft.

Erklärung

Hiermit versichern wir, diese Arbeit
selbständig verfasst und nur die angegebenen
Quellen benutzt zu haben.

(Daniel Beck Jens Müller Frank Ruthardt)