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

Fachstudie Nr. 141

Vergleich von Technologien und Systemen für das Datenmanagement bei wissenschaftlichen Prozessen

Michael Hahn Michael Schneidt

Studiengang: Softwaretechnik

Prüfer: PD. Dr. rer. nat. Holger Schwarz

Betreuer: Dipl.-Inf. Peter Reimann

begonnen am: 20. Juli 2011

beendet am: 17. Februar 2012

CR-Klassifikation: H.2.5, H.2.8, H.4.1

Inhaltsverzeichnis

1 Einleitung **9**

2 Beschreibung der Systeme **11**
2.1 OGSA-DAI . . . 11
2.1.1 Komponenten . . . 12
Ressourcen . . . 12
Aktivitäten . . . 13
Workflows . . . 14
2.1.2 Ausführung von Workflows . . . 15
2.2 SDMCenter . . . 17
2.2.1 Datenmanagement Ebenen . . . 18
2.2.2 Technologien . . . 19
ROMIO . . . 19
Parallel-NetCDF . . . 20
PVFS . . . 20
ADIOS . . . 20
SRM-Lite . . . 20
R . . . 21
ProRata . . . 21
Sapphire . . . 21
FastBit . . . 21
eSimMon . . . 21
Kepler . . . 21
2.3 SIMPL . . . 23
2.3.1 Architektur von SIMPL . . . 23
Eclipse Plug-Ins . . . 24
Apache ODE . . . 24
Apache Axis2 . . . 25
2.3.2 SIMPL Funktionalität und Erweiterbarkeit . . . 25
SIMPL Core . . . 25
Resource Management . . . 27

3 Kriterien für die Evaluation **29**
3.1 Allgemeines . . . 29
3.1.1 Anwendungsbereich . . . 29
3.1.2 Bedienbarkeit, Einfachheit . . . 29
3.1.3 Installation und Administration . . . 29

3.1.4 Abhängigkeiten von anderer Software . . . 30
3.1.5 Dokumentation . . . 30
3.2 Softwarequalität . . . 30
3.2.1 Portabilität/Plattformunabhängigkeit . . . 30
3.2.2 Erweiterbarkeit . . . 30
3.3 Funktionsumfang . . . 31
3.3.1 Datenquellen . . . 31
3.3.2 Datenformate . . . 31
3.3.3 Datenzugriffstechnologien . . . 31
3.3.4 Datenverarbeitungsmöglichkeiten / Datenmanagement-Patterns . . . 32
3.3.5 Flexibilität zur Deploymentzeit . . . 32
3.3.6 Flexibilität zur Laufzeit . . . 33
3.3.7 Möglichkeit Anforderungen an Datenqualität zu formulieren . . . 33
3.3.8 Transparenz der Datenbereitstellung und des Datenmanagements . . . 33
3.4 Leistungsfähigkeit . . . 33
3.4.1 Performanz des Datenzugriffs . . . 34
3.4.2 Performanz der Datenverarbeitung . . . 34
3.4.3 Performanz des Datentransfers . . . 34
3.4.4 Potentielle Optimierungsmöglichkeiten der Datenbereitstellung . . . 34
3.5 Anbindung und Integration . . . 35
3.5.1 Integration von Werkzeugunterstützung . . . 35
3.5.2 Anbindungsmöglichkeiten an andere Systeme/Dienste . . . 35
3.5.3 Fähigkeit verschiedene wissenschaftliche Programme zu koppeln . . . 35

4 Evaluation von OGSA-DAI 37
4.1 Allgemeines . . . 37
4.1.1 Anwendungsbereich . . . 37
4.1.2 Bedienbarkeit, Einfachheit . . . 37
4.1.3 Installation und Administration . . . 38
4.1.4 Abhängigkeiten von anderer Software . . . 39
4.1.5 Dokumentation . . . 40
4.2 Softwarequalität . . . 40
4.2.1 Portabilität/Plattformunabhängigkeit . . . 40
4.2.2 Erweiterbarkeit . . . 40
4.3 Funktionsumfang . . . 42
4.3.1 Datenquellen . . . 42
4.3.2 Datenformate . . . 42
4.3.3 Datenzugriffstechnologien . . . 43
4.3.4 Datenverarbeitungsmöglichkeiten / Datenmanagement-Patterns . . . 44
4.3.5 Flexibilität zur Deploymentzeit . . . 44
4.3.6 Flexibilität zur Laufzeit . . . 45
4.3.7 Möglichkeit Anforderungen an Datenqualität zu formulieren . . . 45
4.3.8 Transparenz der Datenbereitstellung und des Datenmanagements . . . 45
4.4 Leistungsfähigkeit . . . 46
4.4.1 Performanz des Datenzugriffs . . . 46

4.4.2 Performanz der Datenverarbeitung . . . 46
4.4.3 Performanz des Datentransfers . . . 47
4.4.4 Potentielle Optimierungsmöglichkeiten der Datenbereitstellung . . . 48
4.5 Anbindung und Integration . . . 48
4.5.1 Integration von Werkzeugunterstützung . . . 48
4.5.2 Anbindungsmöglichkeiten an andere Systeme/Dienste . . . 48
4.5.3 Fähigkeit verschiedene wissenschaftliche Programme zu koppeln . . . 48

5 Evaluation von SDMCenter 49
5.1 Allgemeines . . . 49
5.1.1 Anwendungsbereich . . . 49
5.1.2 Installation und Administration . . . 50
5.1.3 Bedienbarkeit, Einfachheit . . . 50
5.1.4 Abhängigkeiten von anderer Software . . . 50
5.1.5 Dokumentation . . . 50
5.2 Softwarequalität . . . 51
5.2.1 Portabilität/Plattformunabhängigkeit . . . 51
5.2.2 Erweiterbarkeit . . . 51
5.3 Funktionsumfang . . . 52
5.3.1 Datenquellen . . . 52
5.3.2 Datenformate . . . 52
5.3.3 Datenzugriffstechnologien . . . 52
5.3.4 Datenverarbeitungsmöglichkeiten / Datenmanagement-Patterns . . . 53
5.3.5 Flexibilität zur Deploymentzeit . . . 53
5.3.6 Flexibilität zur Laufzeit . . . 53
5.3.7 Möglichkeit Anforderungen an Datenqualität zu formulieren . . . 53
5.3.8 Transparenz der Datenbereitstellung und des Datenmanagements . . . 53
5.4 Leistungsfähigkeit . . . 54
5.4.1 Performanz des Datenzugriffs . . . 54
5.4.2 Performanz der Datenverarbeitung . . . 54
5.4.3 Performanz des Datentransfers . . . 54
5.4.4 Potentielle Optimierungsmöglichkeiten der Datenbereitstellung . . . 54
5.5 Anbindung und Integration . . . 55
5.5.1 Integration von Werkzeugunterstützung . . . 55
5.5.2 Anbindungsmöglichkeiten an andere Systeme/Dienste . . . 55
5.5.3 Fähigkeit verschiedene wissenschaftliche Programme zu koppeln . . . 55

6 Fazit für SIMPL 57
6.1 Stärken von SIMPL . . . 57
6.1.1 Datenquellen (siehe Kapitel 5.3.1) . . . 57
6.1.2 Datenverarbeitungsmöglichkeiten / Datenmanagement-Patterns (siehe Kapitel 4.3.4 und Kapitel 5.3.4) . . . 57
6.1.3 Flexibilität zur Deploymentzeit (siehe Kapitel 4.3.5 und Kapitel 5.3.5) . . . 58
6.1.4 Flexibilität zur Laufzeit (siehe Kapitel 4.3.6 und Kapitel 5.3.6) . . . 58

6.1.5 Möglichkeit Anforderungen an Datenqualität zu formulieren (siehe Kapitel 4.3.7) und Kapitel 5.3.7 58
6.1.6 Performanz des Datentransfers (siehe Kapitel 5.4.3) 58
6.1.7 Potentielle Optimierungsmöglichkeiten der Datenbereitstellung (siehe Kapitel 4.4.4 und Kapitel 5.4.4) 58
6.1.8 Integration von Werkzeugunterstützung (siehe Kapitel 4.5.1 und Kapitel 5.5.1) 59
6.2 Identifizierte Erweiterungsmöglichkeiten für SIMPL 59
6.2.1 In Bezug auf OGSA-DAI 59
6.2.2 In Bezug auf SDMCenter 60

7 Zusammenfassung und Ausblick **63**

Literaturverzeichnis **65**

Abbildungsverzeichnis

2.1 Datenintegration mithilfe von OGSA-DAI (vgl. [CEMP11]) 11
2.2 Struktur einer Aktivität in OGSA-DAI (vgl. [CEMP11]) 14
2.3 Ausführung eines Workflows mit OGSA-DAI (vgl. [CEMP11]) 17
2.4 Datenmanagement Ebenen vgl. [SDM] 18
2.5 Kepler Workflow Komponenten 22
2.6 SIMPL Architektur 23
2.7 SIMPL Funktionalität und Erweiterbarkeit 26

3.1 Datenmanagement-Pattern-Hierarchie (vgl. [RM11]) 32

4.1 SQLQuery auf mehreren verteilten relationalen Datenbanken mit DQP (vgl. [CEMP11]) 47

Tabellenverzeichnis

4.1 Datenmanagement-Patterns und OGSA-DAI Aktivitäten 44

Verzeichnis der Listings

2.1 Beispiel einer Workflowbeschreibung aus [CEMP11] 16

4.1 Beispiel einer Konfigurationsdatei aus [CEMP11] 39
4.2 Schema des Konfigurationsaufrufs mit Apache Ant aus [CEMP11] 39
4.3 Beispiel einer Data Resource Konfigurationsdatei aus [CEMP11] 41
4.4 Von OGSA-DAI unterstützte Datenquellen aus [CEMP11] 43

1 Einleitung

Im wissenschaftlichen Umfeld werden für Simulationen und Analysen Systeme und Technologien benötigt, mit denen sich Abläufe des Datenmanagements in Form von Prozessen bzw. Workflows modellieren und automatisieren lassen. Dabei ist hauptsächlich wichtig, dass die Systeme und eingesetzten Technologien eine heterogene Datenquellenlandschaft unterstützen bzw. entsprechend erweitert werden können sowie mit großen Datenmengen effizient arbeiten können. Weiterhin ist die Abstraktionsunterstützung für den Anwender wichtig, da dieser meist ein Wissenschaftler oder Ingenieur und daher kein ausgewiesener IT-Experte ist.

Diese Fachstudie evaluiert zwei der frei erhältlichen Datenmanagementsysteme für wissenschaftliche Prozesse nach zuvor festgelegten Kriterien bezüglich wichtiger Eigenschaften und Anforderungen solcher Systeme. Bei den zwei Systemen handelt es sich um *Open Grid Services Architecture Data Access and Integration* (OGSA-DAI, Kapitel 2.1) sowie das *Scientific Data Management Center* (SDMCenter, Kapitel 2.2). Die Evaluierung wird mit dem Hintergrund durchgeführt, dass das vom IAAS und IPVS gemeinsam entwickelte Rahmenwerk SIMPL (SimTech – Information Management, Processes, and Languages), das den Zugriff auf externe Daten in Simulationsworkflows ermöglicht, verbessert werden soll.

Zunächst werden in Kapitel 2 alle Systeme und ihre Funktionsweise beschrieben. Anschließend werden in Kapitel 3 die Kriterien für die nachfolgenden Evaluierungen der Systeme in Kapitel 4 und 5 festgelegt. In Kapitel 6 werden dann, als Fazit, die Vorteile der evaluierten Systeme hinsichtlich der Verbesserungsmöglichkeiten für SIMPL beschrieben und ggf. Umsetzungsvorschläge gemacht, sowie die Vorteile von SIMPL hinsichtlich einiger Kriterien aufgezeigt. Eine Zusammenfassung der Arbeit und einen Ausblick auf weitere Themen liefert abschließend Kapitel 7.

2 Beschreibung der Systeme

In diesem Kapitel werden zwei verschiedene Datenbereitstellungssysteme beschrieben, die im weiteren Verlauf des Dokuments anhand von Kriterien untersucht werden. Weiterhin wird das Datenbereitstellungssystem SIMPL beschrieben, für das durch die Evaluation der zwei anderen Systeme Verbesserungen und Erweiterungen identifiziert werden sollen.

2.1 OGSA-DAI

Open Grid Services Architecture Data Access and Integration (OGSA-DAI) ist ein generisches erweiterbares Rahmenwerk zur Realisierung verteilter Datenzugriffe und verteiltem Datenmanagement. Im Vordergrund steht dabei die Unterstützung von Datenintegration und Datenverarbeitung in Grid- und Cloud-Umgebungen, da gerade dort große, verteilte, heterogene Datenmengen von einer Vielzahl unterschiedlicher Komponenten (Rechner, Services, ...) gleichzeitig genutzt und verarbeitet werden.

Abbildung 2.1 zeigt die Datenintegration mithilfe von OGSA-DAI. OGSA-DAI fungiert dabei als Middleware zwischen Clients (Benutzern) und einer Vielzahl verschiedener realer als auch virtueller Datenquellen. Virtuelle Datenquellen können durch die Integration mehrerer

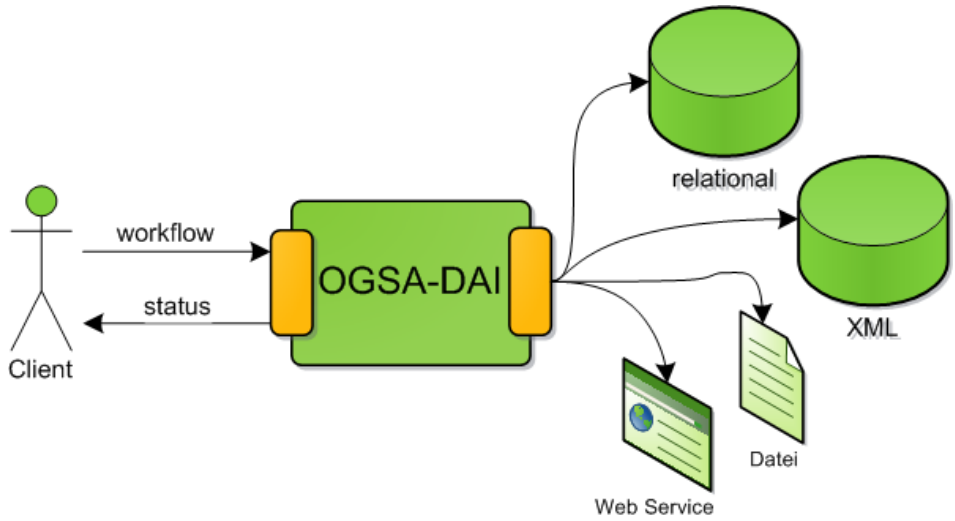

Abbildung 2.1: Datenintegration mithilfe von OGSA-DAI (vgl. [CEMP11])

realer Datenquellen in OGSA-DAI definiert werden. OGSA-DAI unterstützt bereits eine Reihe verschiedener Datenquellen, wie z.B. relationale und XML-Datenbanken, Webservices oder auch strukturierte Dateien. Diese Auswahl kann über einen Erweiterungsmechanismus um weitere Datenquellen ergänzt werden.

In den nachfolgenden Abschnitten wird auf die einzelnen Komponenten von OGSA-DAI und deren Verwendung näher eingegangen. Als Quelle wurde dafür [CEMP11] verwendet.

2.1.1 Komponenten

OGSA-DAI basiert auf drei zentralen Komponenten, mithilfe derer der verteilte Datenzugriff und das Datenmanagement realisiert werden.

Dazu gehören:

Ressourcen Diese abstrahieren konkrete reale Datenquellen in einer für OGSA-DAI kompatiblen Form und können so in Aktivitäten referenziert werden.

Aktivitäten Diese kapseln einzelne konkrete Aufgaben, wie z.B. den Zugriff auf Daten, das Aktualisieren, Kombinieren oder Transformieren von Daten.

Workflows Dies sind komplexe OGSA-DAI Anfragen, die durch strukturiert verbundene Aktivitäten modelliert werden. Listing 2.1 auf Seite 16 zeigt die eXtensible Markup Language (XML)-Repräsentation eines solchen Workflows, der drei verbundene Aktivitäten enthält.

Diese drei Komponenten werden nachfolgend näher beschrieben.

Ressourcen

OGSA-DAI arbeitet nicht direkt mit konkreten physischen Datenquellen, sondern nutzt OGSA-DAI-kompatible Abstraktionen dieser. Diese Abstraktionen werden als OGSA-DAI Ressourcen bzw. Ressourcen bezeichnet. Eine Ressource kapselt dabei beispielsweise nicht nur eine konkrete physische Datenquelle, sondern auch noch weitere zugehörige Daten, wie z.B. Authentifizierungsinformationen, die verwendet werden, um eine physische Verbindung mit der konkreten Datenquelle aufzubauen. Alle Ressourcen besitzen einen eindeutigen logischen Namen, mithilfe dessen die Ressourcen identifiziert und beispielsweise in Aktivitäten referenziert werden können.

OGSA-DAI stellt fünf Typen von Ressourcen bereit, die den Zugriff auf die Funktionalität von OGSA-DAI ermöglichen. Diese fünf Ressourcentypen werden nachfolgend beschrieben.

Data Request Execution Resource (DRER) Eine DRER bildet die zentrale Schnittstelle von OGSA-DAI und initiiert die Ausführung von Anfragen und Workflows. Bei synchronem Aufruf liefert die DRER den Ausführungsstatus an den Client zurück bzw. eine Referenz auf eine Request Resource bei asynchronen Aufruf. Ein OGSA-DAI Server besitzt mindestens eine DRER, um Anfragen und Workflows bearbeiten zu können.

Data Resource Data Resources sind die zentralen Ressourcen in OGSA-DAI. Sie bilden eine OGSA-DAI-spezifische Abstraktion konkreter Datenquellen und ermöglichen dadurch den Zugriff auf physische Datenquellen in OGSA-DAI. Data Resources sind darüber hinaus noch ein wichtiger Erweiterungspunkt in OGSA-DAI, da durch die Bereitstellung neuer Typen von Data Resources weitere Arten von Datenquellen in OGSA-DAI verwendet werden können. So kann beispielsweise mit einer bereits mitgelieferten MySQL-DataResource auf MySQL-Datenbanken aus OGSA-DAI zugegriffen werden.

Data Sink Resource Eine Data Sink Resource ist eine OGSA-DAI Ressource, die Clients erlaubt Daten auf dem Server zu hinterlegen und anderen Clients oder auch Workflows zur Verfügung zu stellen. Datensenken sind eine Möglichkeit für den asynchronen Datenaustausch über OGSA-DAI.

Data Source Resource Eine Data Source Resource erlaubt es Clients hinterlegte Daten von einem OGSA-DAI Server abzufragen. Sie bildet das Gegenstück zur Data Sink Resource zur Realisierung von asynchronem Datenaustausch.

Session Resource Eine Session Resource verhält sich wie ein Statuscontainer, der mit einer Reihe von Workflows verknüpft ist und deren Status für alle enthaltenen Workflows sichtbar macht. So kann der Status eines Workflows mit anderen Workflows geteilt und alle mit der Session Resource assoziierten Workflows korreliert werden. Dadurch lässt sich beispielsweise erreichen, dass ein Workflow B erst startet, sobald ein Workflow A beendet wurde.

Request Resource Eine Request Resource ermöglicht die Verwaltung einer laufenden Anfrage bzw. eines momentan bearbeiteten Workflows. Über die Request Resource kann der Client z.B. den Status der Ausführung oder am Ende deren Ergebniss abfragen oder auch den Workflow ggf. terminieren.

Aktivitäten

Wie schon weiter oben erwähnt, kapseln Aktivitäten einzelne Funktionen bzw. Aufgaben und können so einfach durch Zusammenfügen zu komplexen Datenabfragen (Workflows) kombiniert werden. Eine Aktivität kann dabei das Zugreifen, Aktualisieren, Kombinieren, Transformieren oder Transportieren von Daten umfassen. Aktivitäten können, müssen aber nicht immer, auf Ressourcen zugreifen. So greift beispielsweise die SQLQuery-Aktivität in Abbildung 2.3 auf Seite 17 auf eine MySQL-Data Resource zu, um Daten aus einer Datenbanktabelle auszulesen. Andererseits arbeitet die TupleToWebRowSetCharArrays-Aktivität lediglich auf den Ausgabedaten der SQLQuery-Aktivität und nicht mit den Daten einer Ressource.

Die grundlegende Struktur einer OGSA-DAI Aktivität zeigt Abbildung 2.2. Aktivitäten besitzen dabei immer keine bis mehrere Ein- und Ausgänge und können eine Referenz auf eine Ressource enthalten. Alle Eingänge einer Aktivität müssen immer belegt sein, d.h. entweder mit dem Ausgang einer anderen Aktivität verknüpft oder durch einen Initialwert durch den Client belegt werden. Eine entscheidende Stärke von OGSA-DAI ist der pipeline-artige

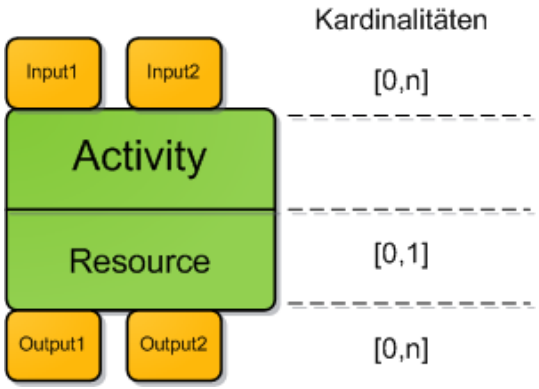

Abbildung 2.2: Struktur einer Aktivität in OGSA-DAI (vgl. [CEMP11])

Datenfluss zwischen den Aktivitäten. So ist es beispielsweise möglich, dass während noch Daten aus einer Ressource gelesen werden, die schon eingelesenen Daten durch nachfolgende Aktivitäten bereits weiter verarbeitet werden. Dies ist möglich, da eine Aktivität immer automatisch startet, sobald alle ihre Eingänge mit einem Wert belegt sind, anschließend die Eingaben verarbeitet, die Ergebnisse an die verknüpften Aktivitäten weiterleitet und so oft diesen Ablauf wiederholt bis keine Eingabedaten mehr vorliegen. Alle nachfolgenden Aktivitäten verhalten sich ebenso und arbeiten so alle auf unterschiedlichen Teilen eines gemeinsamen Datenstroms. Dies führt zu effizienteren Ausführungszeiten und reduziert die Speicherauslastung, da nicht alle Eingabedaten zur selben Zeit im selben Speicherknoten liegen.

Aktivitäten sind ebenfalls ein wichtiger Erweiterungspunkt in OGSA-DAI und ermöglichen beispielsweise die Bereitstellung von benutzerspezifischen Funktionen, die dann in Workflows als Aktivitäten verwendet werden können.

Workflows

Wie schon angesprochen repräsentieren Workflows komplexere OGSA-DAI Anfragen, die durch das Zusammenfügen von Aktivitäten oder auch weiteren Workflows modelliert werden.

Listing 2.1 zeigt wie ein solcher Workflow mithilfe von XML modelliert wird. Darin werden drei Aktivitäten modelliert, die Daten aus einer Datenbank auslesen, diese transformieren und an den Client zurückliefern. Jede der Aktivitäten enthält einen eindeutigen Namen (instanceName) und den Namen der Klasse (name), die die Implementierung bereitstellt. Die SQLQuery-Aktivität enthält weiterhin noch eine Referenz auf die zu verwendende MySQLResource (resource), die die konkrete MySQL-Datenbank kapselt. Als nächstes wird der Datenfluss zwischen den Aktivitäten modelliert. Dazu werden alle initialen Eingabewerte fest zugewiesen (siehe Input von SQLQuery-Aktivität). Der Datenfluss innerhalb des Workflows zwischen den Aktivitäten wird mithilfe von Input- und Output-Streams über Pipes

modelliert. Dazu werden die Ergebnisse der SQLQuery-Aktivität auf einen Output-Stream mit dem Namen „data" auf die Pipe „pipe1" gelegt. Die nachfolgende Aktivität kann die Ergebnisdaten dann mittels eines Input-Streams von derselben Pipe lesen. So werden die Daten zwischen den Aktivitäten weitergeleitet und können auch von mehreren Aktivitäten gleichzeitig genutzt werden.

Alle drei Aktivitäten sind in ein pipeline-Element eingebunden, dies ist der Standard-Typ für Workflows. Es gibt noch zwei weitere Workflow-Typen, die die Ausführungsweise von Workflows beeinflussen, die weitere Workflows enthalten. Nachfolgend werde alle drei Typen aufgezeigt und kurz beschrieben.

Pipeline workflow Ein Pipeline Workflow enthält eine Menge von verketteten Aktivitäten, die alle parallel ausgeführt werden.

Sequence workflow Ein Sequence Workflow enthält eine Menge von Sub-Workflows, die sequentiell ausgeführt werden. Dies wird z.B. verwendet, wenn der erste Sub-Workflow Daten lädt, die im zweiten Sub-Workflow erst nach dem vollständigen Laden verwendet werden können.

Parallel workflow Ein Parallel Workflow enthält eine Menge von Sub-Workflows, die alle parallel ausgeführt werden.

2.1.2 Ausführung von Workflows

In diesem Kapitel wird die Ausführung eines Workflows mit OGSA-DAI anhand eines Beispiels erklärt. Den zugrundeliegenden Beispiel-Workflow zeigt Listing 2.1 auf der nächsten Seite. Abbildung 2.3 auf Seite 17 zeigt alle Komponenten, die für die Ausführung des Workflows von Bedeutung sind und dient als Basis für die nachfolgenden Beschreibungen.

Ein Client schickt den Beispiel-Workflow als XML-Dokument an den Data Request Execution Service (DRES) des OGSA-DAI Servers. Der DRES ist ein Web-Service, der den Zugriff auf einen DRER (siehe Kapitel 2.1.1 auf Seite 12) realisiert. Der DRER sorgt dann für die Ausführung des Workflows, dabei werden folgende Schritte durchlaufen:

1. Einlesen der Workflow-Datei (workflow.xml)
2. Instanziierung aller modellierten Aktivitäten
3. Verknüpfung der Aktivitäten mit den referenzierten Ressourcen
4. Ausführung des Workflows
5. Erstellen eines Request Status
6. Übermittlung des Request Status an den Client

Je nachdem ob der Workflow synchron oder asynchron ausgeführt wird, unterscheidet sich der Ausführungsablauf etwas. Bei synchron ausgeführten Workflows wird, wie oben beschrieben, am Ende der Ausführung ein Request Status generiert und an den Client übermittelt. Der Request Status enhält den Status der Ausführung jeder Aktivität, den Status des gesamten Workflows und eventuell einige Ergebnisdaten. Bei asynchron ausgeführten Workflows wird, direkt nachdem der Workflow gestartet wurde, eine Request Resource

Listing 2.1 Beispiel einer Workflowbeschreibung aus [CEMP11]

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <ns1:request xmlns:ns1="http://ogsadai.org.uk/namespaces/2007/04/types">
3    <ns1:workflow>
4      <ns1:pipeline>
5        <ns1:activity instanceName="SQLQuery" name="uk.org.ogsadai.SQLQuery"
             resource="MySQLResource">
6          <ns1:inputs>
7            <ns1:input name="expression">
8              <ns1:inputLiteral>
9                <ns1:string>SELECT * FROM littleblackbook WHERE id<100</ns1:string>
10             </ns1:inputLiteral>
11           </ns1:input>
12         </ns1:inputs>
13         <ns1:outputs>
14           <ns1:outputStream name="data" pipe="pipe1"/>
15         </ns1:outputs>
16       </ns1:activity>
17       <ns1:activity instanceName="TupleToWRS"
             name="uk.org.ogsadai.TupleToWebRowSetCharArrays">
18         <ns1:inputs>
19           <ns1:input name="data">
20             <ns1:inputStream pipe="pipe1"/>
21           </ns1:input>
22         </ns1:inputs>
23         <ns1:outputs>
24           <ns1:outputStream name="result" pipe="pipe2"/>
25         </ns1:outputs>
26       </ns1:activity>
27       <ns1:activity instanceName="DeliverToRequestStatus"
             name="uk.org.ogsadai.DeliverToRequestStatus">
28         <ns1:inputs>
29           <ns1:input name="input">
30             <ns1:inputStream pipe="pipe2"/>
31           </ns1:input>
32         </ns1:inputs>
33         <ns1:outputs/>
34       </ns1:activity>
35     </ns1:pipeline>
36   </ns1:workflow>
37 </ns1:request>
```

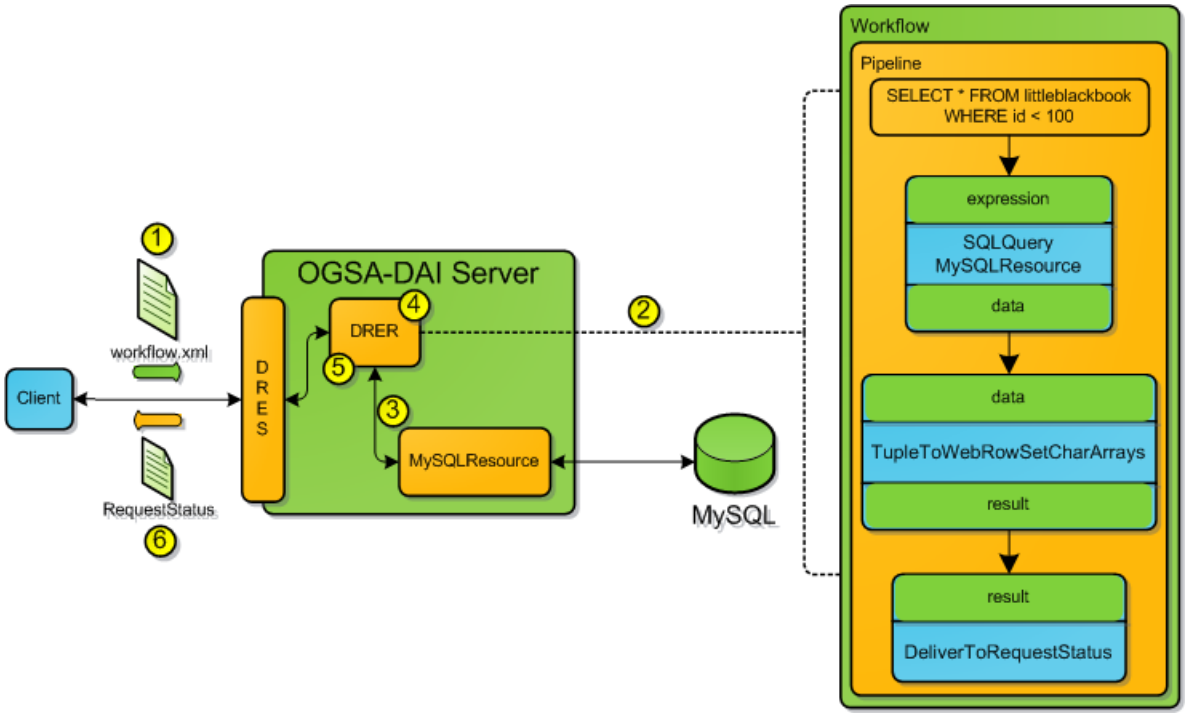

Abbildung 2.3: Ausführung eines Workflows mit OGSA-DAI (vgl. [CEMP11])

(siehe Kapitel 2.1.1 auf Seite 13) und ein Request Status generiert. Der Request Status enthält eine ID, die die Request Resource identifiziert, und wird an den Client zurückgeliefert. Dieser kann nun über den Request Management Service auf die Request Resource zugreifen und den Status der Workflow-Ausführung abfragen oder auch wie bereits beschrieben die Ausführung beispielsweise beenden. Sobald der Workflow beendet wurde, stehen dann dem Client über die Request Resource auch die abschließenden Ergebnisdaten bzw. alle gespeicherten Teilergebnisse, bei vorzeitiger Terminierung durch den Benutzer, des Workflows zur Verfügung.

2.2 SDMCenter

Scientific Data Management Center (SDMCenter) [SDM] ist eine Organisation, die aus dem Scientific Discovery through Advanced Computing (SciDAC) [SCI] Programm des U.S. Department of Energy entstanden ist. Ihr Ziel ist es für das Datenmanagement im wissenschaftlichen Umfeld einen ganzheitlichen Ansatz zu verfolgen und den Wissenschaftler auf allen Ebenen des Datenmanagements zu unterstützen. Dabei stehen vor allem die Performanz und die Skalierbarkeit bei sehr großen Datenmengen im Vordergrund, aber auch eine einfache Modellierung und weitreichende datenverarbeitende und datenanalytische

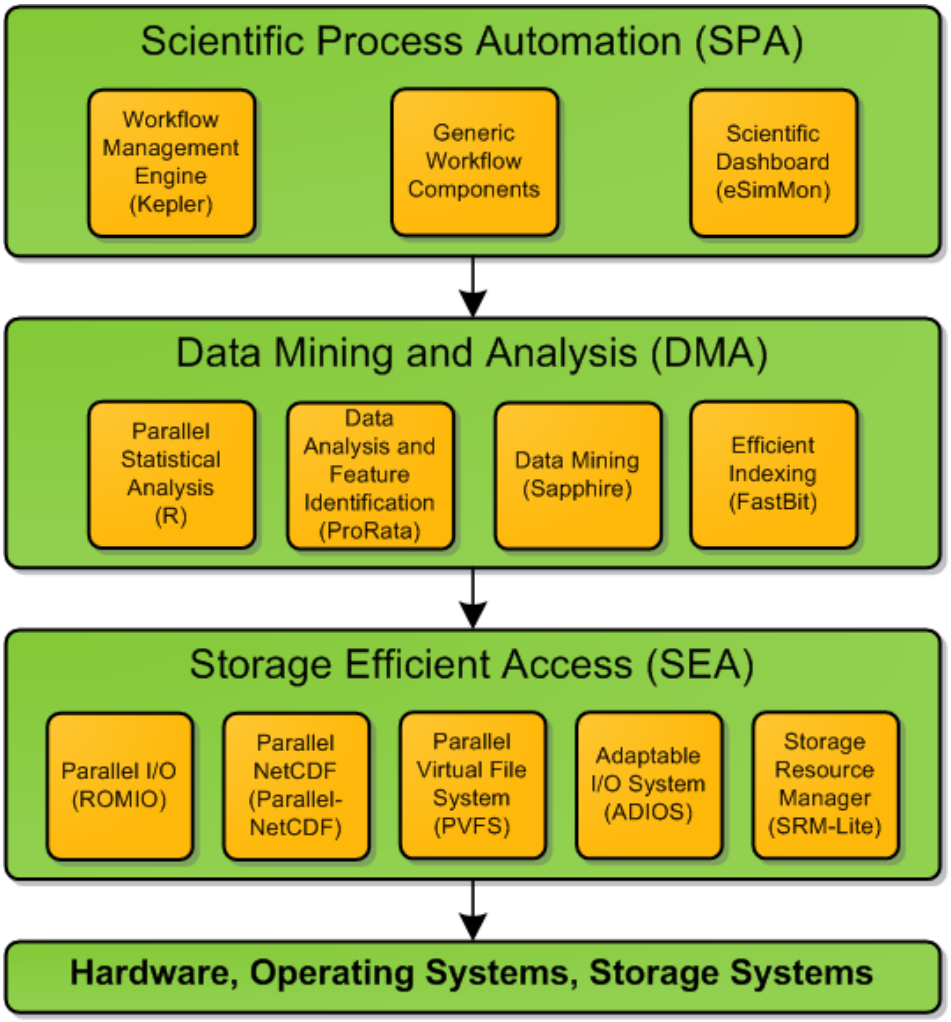

Abbildung 2.4: Datenmanagement Ebenen vgl. [SDM]

Möglichkeiten. In den folgenden Kapiteln werden die verschiedenen Ebenen und die darin angesiedelten Technologien beschrieben.

2.2.1 Datenmanagement Ebenen

Die Anforderungen an das Datenmanagement werden, wie in Abbildung 2.4 dargestellt, von SDMCenter in drei Ebenen unterteilt, die aufeinander aufbauen und in denen jeweils verschiedene Technologien zur Verfügung stehen bzw. entwickelt werden.

Storage Efficient Access Die *Storage Efficient Access* (SEA) Ebene bildet die erste Ebene, die unmittelbar auf der Hardware, Betriebssystemen oder Speichersystemen aufsetzt

und einen effizienten Zugriff auf Daten ermöglicht, ohne dass Simulationen, Visualisierungen oder Analysen dabei beeinträchtig werden. Dafür werden parallele Zugriffstechnologien verwendet, die einen transparenten Zugriff auf die Datenspeicher ermöglichen.

Data Mining and Analysis Die *Data Mining and Analysis* (DMA) Ebene bildet die zweite Ebene und baut auf der SEA-Ebene auf. Sie bietet Technologien für parallele statistische Analysen, Datenanalyse und Data Mining sowie Indexierung von Daten, bei der eine Zuordnung zwischen Daten und Speicherressource erstellt wird, um Daten effizienter zu finden.

Scientific Process Automation Die dritte Ebene ist die *Scientific Process Automation* (SPA) Ebene, in der die Komponenten aus der DMA-Ebene über generische Workflow Komponenten zu einem Workflow zusammengesetzt werden können, der letztendlich auf einer Workflow-Engine automatisiert ausgeführt werden kann. Die Modellierung und Ausführung des Workflows kann dabei auch in wissenschaftliche Dashboards integriert sein, die eine Interaktion zwischen Wissenschaftlern ermöglichen um zum Beispiel Modelle, Daten und Ergebnisse auszutauschen.

2.2.2 Technologien

Jede Ebene bietet verschiedene Technologien, die einzeln aber auch kombiniert in wissenschaftlichen Anwendungen verwendet werden können. Die zentralen Technologien, in Abbildung 2.4 in Klammern stehend, werden in den folgenden Abschnitten kurz beschrieben und ihre Funktionsweise erklärt. Auf die relevanten Technologien für diese Fachstudie wird im späteren Verlauf noch detaillierter eingegangen.

ROMIO

ROMIO [ROM] ist eine High-Performance Implementierung des Message-Passing Interface Standards (MPI) [MPIa], der den Datenaustausch über Nachrichten zwischen parallelen Prozessen in verteilten Systemen beschreibt. Dabei handelt es sich um eine Programmbibliothek, die für verschiedene Plattformen, Betriebssysteme und Dateisysteme für die Programmiersprachen C und Fortran zur Verfügung steht. ROMIO unterstützt Linux Betriebssysteme und verschiedene Hardware-Plattformen. Neuere Versionen sind allerdings nicht mehr als unabhängiges Paket verfügbar, sondern werden als Teil von MPICH2 [MPIb] ausgeliefert. MPICH2 unterstützt auch andere Betriebssysteme und weitere Plattformen wie Computer-Cluster und High-Speed-Netzwerke, außerdem enthält es ein Binding für die Programmiersprache C++.

Parallel-NetCDF

Parallel-NetCDF [PNE] ist eine Bibliothek, die einen hochperformanten Zugriff auf Dateien im NetCDF-Format ermöglicht. NetCDF [NET] ist ein selbstbeschreibendes, maschinenunabhängiges Dateiformat für den Austausch von wissenschaftlichen Daten. Die NetCDF-Programmbibliothek ist allerdings nur für den seriellen Zugriff ausgelegt und verhindert damit einen hochperformanten Zugriff, der nur parallel erreicht werden kann. Parallel-NetCDF erweitert die NetCDF-Programmbibliothek dahingehend, dass eine MPI-Implementierung mit I/O-Unterstützung (MPI-IO) wie z.B. ROMIO verwendet werden kann, um die Zugriffe zu parallelisieren.

PVFS

PVFS [PVF] ist ein virtuelles, hochperformantes, ausfallsicheres und hardwareunabhängiges Dateisystem, das dafür ausgelegt ist riesige verteilte Datenmengen im Petabyte-Bereich unter sehr hohen Zugriffsraten zu bewältigen. Es unterstützt außerdem ROMIO für den direkten Datenzugriff ohne Umweg über das Betriebssystem und lässt sich leicht installieren. Allerdings kann es nur unter Linux eingesetzt werden.

ADIOS

ADIOS [ADI] ist ein anpassungsfähiges IO-System, das dem Wissenschaftler eine einfache und flexible Verarbeitung von Daten in Simulationsprogrammen ermöglicht, ohne dass bestehender Programmcode geändert werden muss. Das System ist für die Programmiersprachen C und Fortran erhältlich und stellt dafür entsprechende Standard-IO-Routinen zur Verfügung, die im Programmcode verwendet werden können. Die Funktionsweise der Routinen, wie zum Beispiel die synchrone oder asynchrone Verarbeitung, kann dann, je nach Anforderung an die Simulation, in einer externen XML-Datei konfiguriert werden. Die Konfiguration bietet dabei einen sehr hohen Detailgrad und ermöglicht die Kontrolle auf Basis von Datenelementen und Datentypen. Dadurch können Anwendungen einfach und flexibel an verschiedene Infrastrukturen angepasst werden, ohne Änderungen am Programmcode vorzunehmen.

SRM-Lite

SRM-Lite [SRM] ist ein leichtgewichtiger, kommandozeilenbasierter Storage Resource Manager, der die Protokolle http, https, ftp, gridftp, srm und scp zur Datenübertragung unterstützt. Damit können Daten zuverlässig und sicher zwischen verschiedenen Speicherorten bewegt werden. Eine Verwaltung der Datenquellen bzw. Metdaten der Datenquellen wird von SRM-Lite nicht unterstützt.

R

R [RPR] ist ein System für statistische Berechnungen, Datenbearbeitung und grafische Darstellung von Daten. Es besitzt eine eigene Programmiersprache und ein Kommandozeileninterface und steht für Linux und Windows zur Verfügung.

ProRata

ProRata [PRO] ist eine Software zur quantitativen Datenanalyse der Proteinzusammensetzung des Proteoms, der Gesamtheit aller Proteine eines Lebewesens.

Sapphire

Sapphire [SAP] ist eine Software für das Data Mining bei komplexen, mehrdimensionalen wissenschaftlichen Daten. Dafür werden Konzepte des Data Mining, der Video- und Bildverarbeitung und der Mustererkennung angewendet und skalierbare Algorithmen entwickelt.

FastBit

FastBit [FAS] ist eine Open Source Bibliothek zur Datenverwaltung, die das NoSQL-Konzept auf Grundlage komprimierter Bitmap Indexe verfolgt und damit eine sehr schnelle Suche von Daten ermöglicht.

eSimMon

eSimMon [ESS] ist ein kollaboratives webbasiertes System für das Monitoring und die Analyse von Simulationen. Die Simulationsdaten können über eine PHP-API in das System eingespeist werden, es ist aber auch möglich das System über eine C-API direkt mit einer Simulation zu verbinden. Das System basiert auf Adobe Flash [ADO], speziell wegen der Möglichkeiten zur grafischen Darstellung von Vektorgrafiken und dreidimensionalen Inhalten.

Kepler

Kepler [KEP] ist ein Open Source Workflow Management System für die Modellierung und Analyse von wissenschaftlichen Daten, das Wissenschaftlern und Analysten beim interdisziplinären Austausch von Daten, Modellen und Analysen unterstützen soll. Das System basiert auf Java und besteht aus einer Workflow Engine, einer grafischen Benutzeroberfläche (GUI) sowie einem Kommandozeileninterface und benutzt R zur Datenanalyse und grafischen Darstellung von Daten.

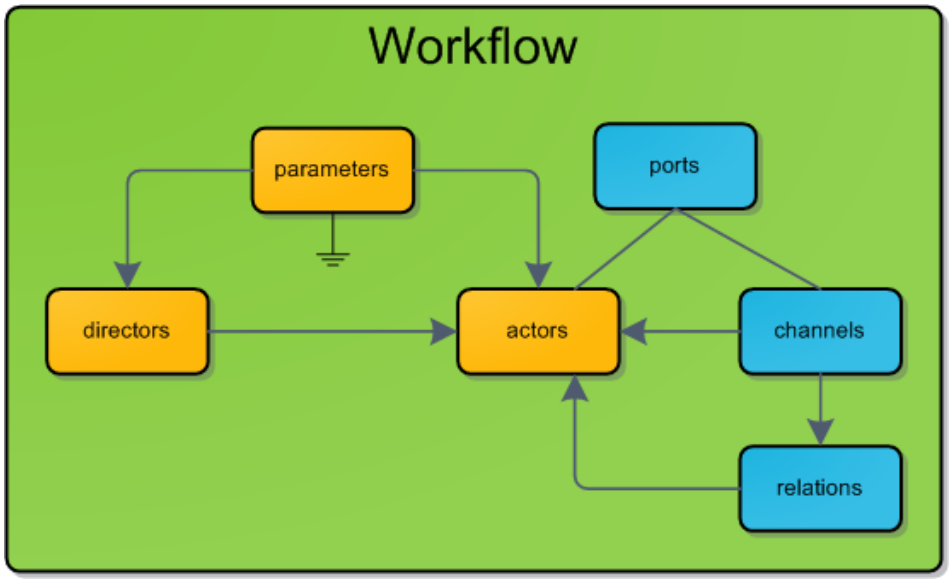

Abbildung 2.5: Kepler Workflow Komponenten

Kepler besitzt bereits eine Bibliothek mit über 350 einsatzfertigen Workflow-Komponenten, die aber beliebig um eigene Komponenten erweitert werden kann, die auch mit anderen Benutzern geteilt werden können. Die verschiedenen Workflow-Komponenten von Kepler und ihre Beziehungen werden in Abbildung 2.5 dargestellt. Es gibt drei verschiedene Arten von Workflow-Komponenten: Direktoren (directors), Aktoren (actors) und Parameter (parameters). Die Aktoren können über Anschlüsse (ports) und Kanäle (channels) sowie Verbindungen (relations) miteinander in Beziehung gebracht werden können. Direktoren steuern den Ablauf des Workflows und teilen den Aktoren mit, wann sie in Aktion treten sollen, ob sie beispielsweise sequentiell oder parallel ausgeführt werden. Aktoren übernehmen dagegen die eigentlichen Aufgaben des Datenmanagements, der Datenverarbeitung, der Datenanalyse oder der Darstellung von Daten. Die Daten werden über Kanäle als Tokens übertragen und können über Verbindungen an mehrere Aktoren verteilt werden. Parameter sind konfigurierbare Werte, die an Workflows, an Aktoren oder an Direktoren angehängt werden können und deren Verhalten beeinflussen. Es gibt bereits Aktoren für den Zugriff auf verschiedene Datenquellen wie z.B. Datenbanken, Dateisysteme und Grid Systeme, und es werden verschiedene Datenformate unterstützt, insbesondere die Ecological Metadata Language (EML) [EML]. Außerdem gibt es einen Web Service Aktor, um Daten von einem Web Service abzurufen und weitere XML-verarbeitende Aktoren, um die angeforderten Daten zu verarbeiten.

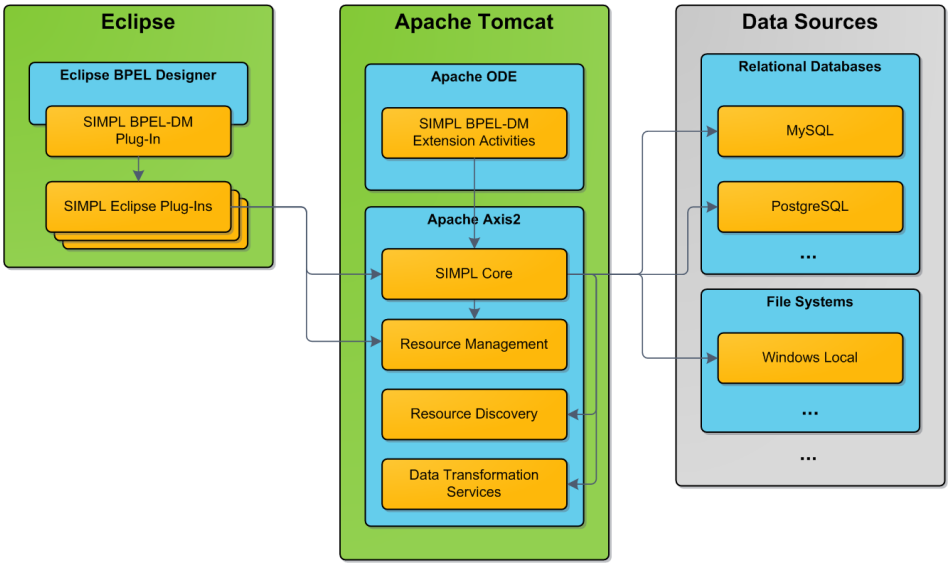

Abbildung 2.6: SIMPL Architektur

2.3 SIMPL

In diesem Kapitel wird das Rahmenwerk SIMPL (SimTech - Information Management, Processes and Languages) beschrieben, das die Möglichkeit bietet aus BPEL (Business Process Execution Language)-Workflows auf beliebige Datenquellen zuzugreifen.

Zunächst wird ein Überblick über die Architektur von SIMPL gegeben und anschließend auf das Datenmanagement sowie die Funktionsweise und Erweiterungsmöglichkeiten des *SIMPL Core* eingegangen, der den Zugriff auf die Datenquellen ermöglicht.

2.3.1 Architektur von SIMPL

Die Architektur von SIMPL, die in Abbildung 2.6 dargestellt ist, besteht zum einen aus der Modellierungsumgebung *Eclipse*, in der Workflows mit Datenquellenzugriffen modelliert werden können, und zum anderen aus der Laufzeitumgebung *Apache Tomcat*, mit der die modellierten Workflows ausgeführt werden können und in der der Zugriff auf die Datenquellen (*Data Sources*) realisiert wird. SIMPL erweitert dazu Eclipse für die Modellierung um Datenmanagement-Aktivitäten und die Laufzeitumgebung für die Ausführung dieser Aktivitäten. Die Laufzeitumgebung *Apache Tomcat* teilt sich dabei in die zwei Bereiche *Apache ODE* und *Apache Axis2*, die beide ein Container für Web Services bereitstellen bzw. darstellen. Es ist daher grundsätzlich möglich die Komponenten, die unter *Apache Axis2* dargestellt sind, im *Apache ODE* Container als Web Service, sowie direkt im Klassenpfad von *Apache ODE* bereitzustellen, um die Performance bei der Interaktion dieser Komponenten zu erhöhen. Mit der Darstellung soll aber klar gezeigt werden, dass es sich bei den Komponenten unter

Apache Axis2 um grundsätzlich unabhängige Komponenten zu den Komponenten in *Apache ODE* handelt, die auch in anderen Laufzeitumgebungen als Web Service bereitgestellt werden können.

Eclipse Plug-Ins

SIMPL besteht aus den folgenden Eclipse Plug-Ins.

SIMPL BPEL-DM Plug-In Das *SIMPL BPEL-DM Plug-In* erweitert den Eclipse BPEL Designer um zusätzliche Datenmanagement-Aktivitäten, mit denen Zugriffe auf Datenquellen modelliert werden können. Jede Aktivität definiert dabei eine Datenquelle und einen Befehl, in der von der Datenquelle unterstützten Befehlssprache. Bei der Ausführung der Aktivität, wird der Befehl an die Datenquelle geschickt, wo er ausgeführt wird. Die Datenmanagement-Aktivitäten umfassen den Abruf von Daten, das Zurückschreiben von Daten, den Transfer von Daten und das Absetzen von Datenmanipulations- und Datendefinitions-Befehlen auf der Datenquelle und können um beliebige weitere Aktivitäten erweitert werden.

SIMPL Eclipse Plug-Ins Die *SIMPL Eclipse Plug-Ins* sind mehrere Plug-Ins für die Kommunikation zwischen Eclipse und den SIMPL Komponenten sowie die Konfiguration dieser Komponenten. Dazu gehören z.B. eine erweiterbare Adminkonsole für die Konfiguration des *SIMPL Core*, zusätzliche Eclipse Einstellungen für die Adressen der SIMPL Web Services und eine Übersicht der verfügbaren Datenquellen im Resource Management.

Apache ODE

Die BPEL-Engine *Apache ODE* ist für die Ausführung der Workflows zuständig und unterstützt das BPEL-Erweiterungskonzept der Extension Activities, mit denen eigene Aktivitäten in Workflows definiert werden können. Die in *Apache ODE* laufenden Komponenten von SIMPL werden in den folgenden Abschnitten beschrieben.

SIMPL BPEL-DM Extension Activities Die *SIMPL BPEL-DM Extension Activities* implementieren die Ausführungslogik für die vom *SIMPL BPEL-DM Plug-In* definierten Datenmanagement-Aktivitäten, damit diese zur Laufzeit erkannt und ausgeführt werden können. Das Senden der Befehle an die Datenquellen läuft dabei über den *SIMPL Core*.

SIMPL Core Der *SIMPL Core* ist die Kernkomponente von SIMPL. Über ihn laufen alle Zugriffe auf Datenquellen und er lässt sich durch ein Plug-In-Schnittstellen (Extension Points) um beliebige Datenquellen und Datenquellen-Typen erweitern. Eine genauere Beschreibung des *SIMPL Core* und sein Zusammenspiel mit den Komponenten *Resource Management*, *Resource Discovery* und *Data Transformation Services* liefert das Kapitel 2.3.2.

Apache Axis2

Apache Axis2 (Apache eXtensible Interaction System 2) ist ein Web Service Framework, mit dem weitere SIMPL-Komponenten als Web Services bereitgestellt werden, die in den folgenden Abschnitten beschrieben werden.

Resource Management Das *Resource Management* verwaltet zentral Metadaten zur Beschreibung aller Ressourcen von SIMPL in einer PostgreSQL Datenbank. Zu den Ressourcen gehören die zur Verfügung stehenden Datenquellen, Plug-Ins und Services.

Resource Discovery SIMPL unterstützt auch das Late Binding von Datenquellen, die zur Laufzeit noch nicht festgelegt sind. Es können damit bei Datenquellen Eigenschaften hinterlegt werden, die erst zur Laufzeit mit den vom Workflow-Modellierer festgelegten Anforderungen verglichen werden, um eine passende Datenquelle ausfindig zu machen. Für die Suche nach einer passenden Datenquelle ist die *Resource Discovery* zuständig, die dafür verschiedene Strategien unterstützt wie zum Beispiel das Finden des ersten Ergebnisses bei kompletter Übereinstimmung zwischen Datenquellen-Eigenschaften und festgelegten Anforderungen.

Data Transformation Services SIMPL verwendet für unterschiedliche Datenquellentypen unterschiedliche XML-Datenformate, um abgerufene Daten im Workflow verarbeitungsfähig zu machen. Mit den *Data Transformation Services* ist es außerdem möglich zwischen diesen verschiedenen Datenformaten zu konvertieren und somit Daten zwischen Datenquellen unterschiedlichen Typs auszutauschen. Damit ist es z.B. möglich Daten von einer relationalen Datenbank abzurufen und in eine Datei auf ein Dateisystem zu schreiben.

2.3.2 SIMPL Funktionalität und Erweiterbarkeit

Dieses Kapitel beschreibt die Funktionalität und Erweiterbarkeit von SIMPL und das Zusammenspiel der beteiligten Komponenten (siehe Abbildung 2.3.2).

SIMPL Core

Der *SIMPL Core* besteht aus verschiedenen Diensten (Services) und Erweiterungspunkten (Extension Points), die in den folgenden Abschnitten beschrieben werden.

SIMPL Core Service Der *SIMPL Core Service* ist für den Zugriff auf Datenquellen zuständig und besitzt Extension Points, über die der Zugriff um weitere Datenquellen erweitert werden kann.

Über den *Connector Extension Point* können Connector Plug-Ins für verschiedene Datenquellentypen und Anfragesprachen erstellt werden, die die direkte Verbindung zu den Datenquellen herstellen und darauf Operationen wie z.B. den Abruf von Daten ausführen können. Damit abgerufene Daten in einem Workflow weiterverarbeitet

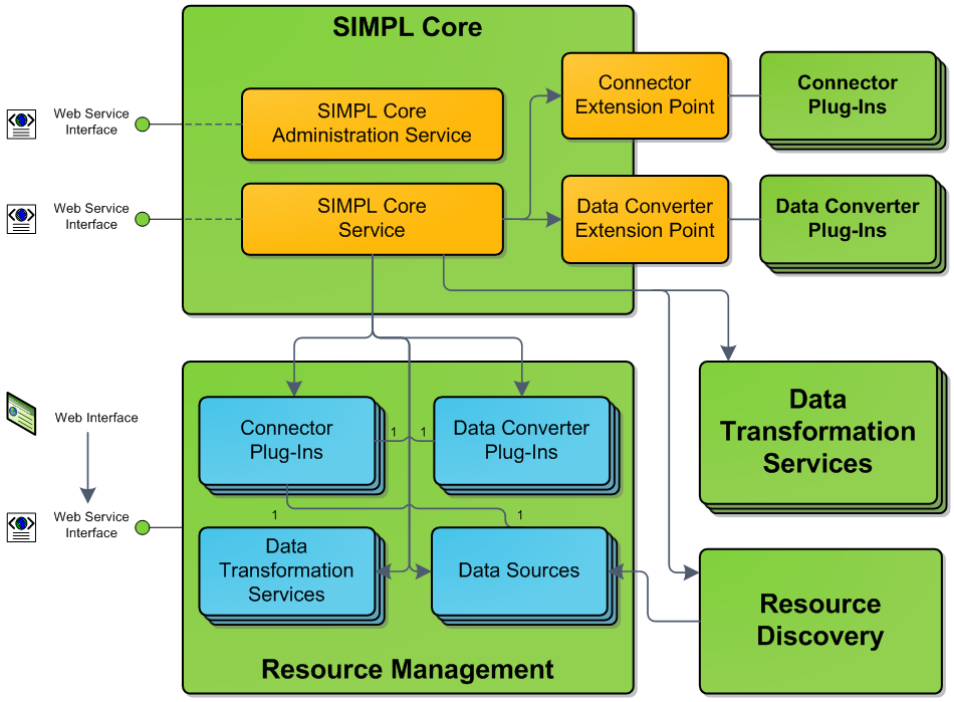

Abbildung 2.7: SIMPL Funktionalität und Erweiterbarkeit

werden können, müssen die Daten vom Connector-Datenformat, wie z.B. ein JDBC-ResultSet, in ein XML-Datenformat konvertiert werden. Ebenso müssen Daten, die aus dem Workflow in eine Datenquelle geschrieben werden, zuvor in ein entsprechendes Connector-Datenformat konvertiert werden, wie z.B. eine Liste von SQL Statements, die entsprechende Änderungs- bzw. Einfüge-Operationen in einer Datenbank durchführen. Zur Realisierung dieser Datenformatkonvertierungen können über den *Data Converter Extension Point* Data Converter erstellt werden, die auch von mehreren Connectoren verwendet werden können.

Falls Daten zwischen verschiedenen Datenquellentypen ausgetauscht werden, kann der *SIMPL Core Service* zusätzlich auf *Data Transformation Services* zurückgreifen, um zwischen den verschiedenen XML-Datenformaten zu konvertieren, die von den entsprechenden Data Convertern verstanden werden.

Die Datenquellen werden im Workflow beim statischen Binden über einen eindeutigen Namen referenziert, der vom *SIMPL Core Service* zur Laufzeit über das *Resource Management* aufgelöst wird. Beim dynamischen Binden, dem Late Binding, wendet sich der *SIMPL Core Service* an die *Resource Discovery*, um Datenquellen ausfindig zu machen, die den Anforderungen genügen.

Über ein *Web Service Interface* steht der *SIMPL Core Service* auch anderen Anwendungen oder Workflow-Engines zur Verfügung.

SIMPL Core Administration Service Der *SIMPL Core Administration Service* verwaltet die internen Einstellungen des *SIMPL Core* in einer eingebetteten Datenbank. Er besitzt außerdem ein *Web Service Interface*, über die die Einstellungen von der Adminkonsole (siehe Kapitel 2.3.1) in Eclipse abgerufen und gespeichert werden können.

Resource Management

Im *Resource Management* werden die verfügbaren *Connector Plug-Ins*, *Data Converter Plug-Ins* des *SIMPL Core Service* sowie *Data Transformation Services* und Datenquellen registriert und verwaltet. Dabei wird jedem Connector ein passender Data Converter zugewiesen und jeder Datenquelle ein passender Connector. Die SIMPL Komponenten können benötigte Resourcen direkt auf Klassenebene vom *Resource Management* abrufen, sowie über ein *Web Service Interface*. Für die Verwaltung durch einen Administrator gibt es zusätzlich ein *Web-Interface*.

3 Kriterien für die Evaluation

In diesem Kapitel werden die Kriterien definiert und beschrieben, die zur Evaluation und dem Vergleich der Systeme verwendet werden.

3.1 Allgemeines

Dieser Abschnitt enthält alle Kriterien, mit denen allgemeine Anforderungen an die Systeme untersucht werden.

3.1.1 Anwendungsbereich

In diesem Punkt wird geprüft, für welchen Anwendungsbereich das untersuchte System entworfen wurde und weswegen es sich dafür besonders eignet.

3.1.2 Bedienbarkeit, Einfachheit

Eine einfache und intuitive Bedienung sowie ein geringer Einarbeitungsaufwand stellen zentrale Punkte in der Verwendung von Software dar. Im Zusammenhang mit Datenbereitstellungsystemen ist dies besonders wichtig, da diese von einer Vielzahl von Benutzergruppen für die unterschiedlichsten Anwendungsfälle eingesetzt werden.

3.1.3 Installation und Administration

Die Komplexität und der Aufwand der Installation und Administration sind entscheidende Kriterien für die Akzeptanz und Verbreitung eines Systems. Dabei sind beispielsweise die Möglichkeit einer automatischen Installation, die Komplexität einer manuellen Installation oder die Qualität der Installationsanleitung entscheidend. Ebenso gilt es zu prüfen, wie komplex bzw. aufwendig die Administration der Systeme ist und ob der Nutzer dabei z.B. durch eine entsprechende Benutzeroberfläche unterstützt wird.

3.1.4 Abhängigkeiten von anderer Software

In diesem Punkt wird betrachtet, ob das untersuchte System Abhängigkeiten von weiterer Software besitzt und inwieweit sich diese auf die Verwendung des Systems auswirken.

3.1.5 Dokumentation

Eine fehlende oder unzureichende bzw. qualitativ schlechte Dokumentation schränkt den Benutzer in der Verwendung des Systems stark ein und erschwert oder verhindert sogar den Einstieg in die Verwendung des Systems. Die Dokumentation spielt vor allem beim Ausschöpfen des vollen Potentials eines Systems oder auch bei einer Erweiterung des Systems um mögliche benutzerspezifische Komponenten eine entscheidende Rolle, da diese ohne Dokumentation nur schwer realisierbar sind.

3.2 Softwarequalität

Dieser Abschnitt beschreibt einige Kriterien, die sich mit der Qualität der Softwaresysteme befassen.

3.2.1 Portabilität/Plattformunabhängigkeit

Gerade im wissenschaftlichen Bereich ist die Plattformunabhängigkeit und damit auch die Portabilität ein wichtiges Kriterium, da viele wissenschaftliche Softwaresysteme an eine bestimmte Plattform gebunden sind und damit gewissen Einschränkungen unterliegen. Durch ein plattformunabhängiges und portables Datenbereitstellungssystem kann zumindest die Datenbereitstellung über verschiedene Plattformen hinaus vereinheitlicht werden. Dies wirkt sich auch auf die Verbreitung und Akzeptanz des Systems aus, da es auf einheitliche Weise in verschiedenen Umgebungen verwendet werden kann.

3.2.2 Erweiterbarkeit

In diesem Punkt wird geprüft, wie erweiterbar ein System ist. Der Fokus liegt dabei auf der Erweiterbarkeit des Systems in Bezug auf die Anbindung weiterer Datenquellen, die Definition weiterer Datenformate oder auch die Erweiterung der Datenverarbeitung. Dabei spielt es auch eine Rolle, wie genau die Erweiterungen realisiert werden können, welche Anforderungen dafür an den Nutzer gestellt werden und mit welchem Aufwand dies verbunden ist.

3.3 Funktionsumfang

In diesem Abschnitt wird auf Kriterien im Zusammenhang mit dem Funktionsumfang der Systeme eingegangen.

3.3.1 Datenquellen

Die Funktionalität zur Anbindung von möglichst vielen verschiedenen, heterogenen Datenquellen bildet den Kern eines Datenbereitstellungssystems. Dabei ist nicht nur die Anzahl der unterstützten Datenquellen oder Datenquellentypen zu beachten, sondern auch deren Integrationsgrad. D.h. inwieweit auf datenquellenspezifische Funktionen zugegriffen oder besondere Merkmale der jeweiligen Datenquelle ausgenutzt werden können. Dabei ist ebenfalls zu prüfen, wie (flexibel) die Datenquellen konkret an das System angebunden werden und wie die Authentifizierung abgewickelt wird.

3.3.2 Datenformate

In diesem Punkt wird untersucht, welche Datenformate das System in welchem Umfang unterstützt. Dabei sollen alle Arten von Datenformaten betrachtet werden. Dies sind z.B. interne Datenformate des Systems, lokale Datenformate der unterstützten Datenquellen oder auch Datenformate, die für die Übertragung von Daten genutzt werden. Hierbei ist ebenfalls zu untersuchen inwieweit die (automatische) Transformation zwischen verschiedenen Datenformaten möglich ist und welche Einschränkungen dabei eventuell gelten.

3.3.3 Datenzugriffstechnologien

Die Unterstützung verschiedener Datenzugriffstechnologien erweitert bzw. verringert gleichermaßen die Palette der unterstützten Datenquellen. So sichert beispielsweise die Verwendung der Java Database Connectivity (JDBC) API in Java bzw. der Open Database Connectivity (ODBC) API in C++ die Anbindung aller Datenquellen, die einen entsprechenden Treiber für diese APIs bereithalten. Eine andere Datenzugriffstechnologie bilden z.B. Web Services, die eine programmiersprachen- und plattformunabhängige Schnittstelle bereitstellen, über die Daten abgerufen, verarbeitet oder gespeichert werden können. In der konkreten Implementierung der Web Services wird der Datenzugriff dann in einer konkreten Programmiersprache mithilfe entsprechender APIs (z.B. JDBC) realisiert, dies ist aber nach außen durch die einheitliche Schnittstelle nicht sichtbar.

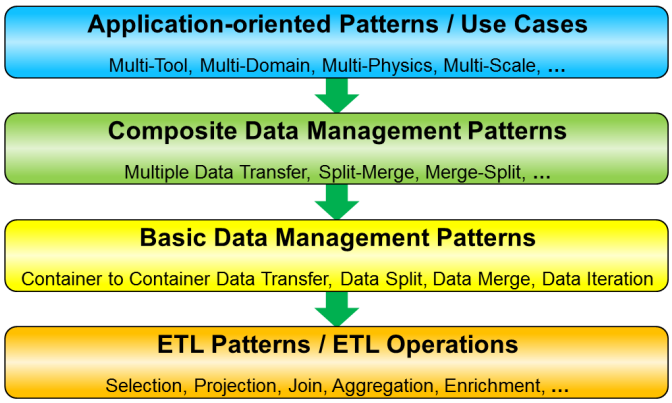

Abbildung 3.1: Datenmanagement-Pattern-Hierarchie (vgl. [RM11])

3.3.4 Datenverarbeitungsmöglichkeiten / Datenmanagement-Patterns

Abbildung 3.1 zeigt eine Hierarchie von Datenmanagement-Patterns, die typische Datenverarbeitungs- bzw. Datenbereitstellungsschritte in Simulationsprozessen darstellt. In diesem Punkt wird geprüft, welche Ebenen dieser Hierarchie in welchem Umfang durch die untersuchten Systeme unterstützt werden. Die Ebene „ETL Patterns / ETL Operations" beinhaltet Extract, Transform, Load (ETL) -Patterns, die Bestandteile von ETL-Prozessen sind, mit denen Daten aus mehreren unterschiedlichen Datenquellen *extrahiert*, diese dann in das Datenformat der Zieldatenquelle *transformiert* und schließlich in die Zieldatenquelle *geladen* werden können. Typische Patterns aus dieser Ebene sind Selektion (*Selection*), Projektion (*Projection*), Verbund (*Join*) sowie die Aggregation (*Aggregation*) und das Anreichern von Daten (*Enrichment*). Die Ebene „Basic Data Management Patterns" setzt auf der ETL-Ebene auf und enthält abstraktere Patterns, wie beispielsweise das Verschieben von Daten zwischen zwei Datencontainern (*Container-to-Container Data Transfer*), das Auftrennen (*Data Split*) und Zusammenfügen von Daten (*Data Merge*) sowie das Iterieren über Daten (*Data Iteration*). Die nächst höhere Ebene „Composite Data Management Patterns" enthält Patterns, die mehrere Patterns der „Basic Data Management Patterns"-Ebene miteinander verknüpfen. So wird beispielsweise durch die Verknüpfung der Patterns *Data Split* und *Data Merge* ein neues Pattern *Split-Merge*. Die Ebene „Application-oriented Patterns / Use Cases" bildet die höchste Ebene der Hierarchie und stellt anwendungsspezifische Patterns bereit, die aus Kombinationen von Patterns der darunterliegenden Ebenen bestehen. Dazu gehören im wissentschaftlichen Bereich beispielsweise *Multi-Tool*, *Multi-Domain*, *Multi-Physics* und *Multi-Scale* Anwendungsfälle.

3.3.5 Flexibilität zur Deploymentzeit

Die Flexibilität zur Deploymentzeit bestimmt die Möglichkeiten, die ein Benutzer hat, um zur Deploymentzeit Einfluss auf einen Prozess zu nehmen. Dazu gehört beispielsweise die

Möglichkeit eine bestimmte Datenquelle für den Datenzugriff beim Deployment festzulegen, die bei der Modellierung noch nicht bekannt ist.

3.3.6 Flexibilität zur Laufzeit

Die Flexibilität zur Laufzeit bestimmt die Möglichkeiten, die ein Benutzer hat, um zur Laufzeit Einfluss auf einen Prozesses zu nehmen. Dazu gehört beispielsweise die Möglichkeit eine Datenquelle über bei der Modellierung formulierte Anforderungen erst zur Laufzeit ausfindig zu machen.

3.3.7 Möglichkeit Anforderungen an Datenqualität zu formulieren

In diesem Punkt wird geprüft, inwieweit es dem Nutzer möglich ist Anforderungen an die Qualität von Daten zu stellen und inwieweit diese Anforderungen auch evaluiert werden. Ein Beispiel hierfür wäre eine Anforderung an die Aktualität der Daten, d.h. der Nutzer kann zusammen mit der Datenquellen-Anfrage formulieren, wie „frisch" die gelieferten Daten sein müssen, damit sie verwendet werden können. Weitere Informationen dazu liefert [RBD+11].

3.3.8 Transparenz der Datenbereitstellung und des Datenmanagements

In diesem Punkt wird geprüft, inwieweit die Systeme den Nutzer beim Überwachen (Monitoring), beim Nachvollziehen (Provenance) und beim Protokollieren (Auditing) der ausgeführten Datenbereitstellungs- und Datenverarbeitungs-Operationen unterstützen. Beim Auditing werden dabei nur die reinen Ereignisdaten protokolliert. Für die Provenance werden zusätzliche Information, wie beispielsweise der Zustand einer Datenquelle, welche Operationen auf welchen Daten konkret ausgeführt wurden und welche Datenbewegungen zwischen welchen Ressourcen stattgefunden haben, protokolliert. Alternativ können die zusätzlichen Informationen für die Provenance auch auf explizitem oder implizitem Wissen beruhen. Dieses Wissen kann z.B. direkt aus den ausgeführten Prozessen gewonnen werden, wenn darin die Operationen und die Informationen, auf welche Datenquellen wann und wie zugegriffen wird, direkt sichtbar sind. Das Monitoring sorgt für die Aufbereitung der Auditing-Daten in eine für den Nutzer verständliche Form. Gerade bei komplexen Operationen auf eventuell mehreren Datenquellen ist es für die Fehlersuche nahezu unumgänglich, dass zumindest die Ereignisdaten dem Nutzer zur Verfügung stehen.

3.4 Leistungsfähigkeit

Dieser Abschnitt beschreibt einige Kriterien, die sich mit der Leistungsfähigkeit der Systeme befassen.

3.4.1 Performanz des Datenzugriffs

In diesem Punkt wird untersucht, wie performant der Datenzugriff durch die Systeme realisiert ist. Kriterien dafür sind beispielsweise die Anzahl der physischen Zugriffe auf die Datenquelle während der Ausführung einer Anfrage, ob für jeden Datenquellenzugriff eine neue physische Verbindung zur Datenquelle hergestellt und dann wieder verworfen wird oder ob Verbindungen in einem Connection-Pool gehalten und wiederverwendet werden.

3.4.2 Performanz der Datenverarbeitung

Hier wird festgestellt, wie performant die Datenverarbeitungsmöglichkeiten sind, ob zum Beispiel die Verarbeitung bereits auf Datenquellenseite stattfindet oder zur Verarbeitung mehr Daten als nötig in das zu untersuchende System übertragen werden.

3.4.3 Performanz des Datentransfers

Beim Transfer von großen Datenmengen gibt es oft die Möglichkeit Referenzen auf die Daten zu verschicken, die nur an Stellen aufgelöst werden, an denen die Daten konkret benötigt werden. Oder es gibt Möglichkeiten die Daten komprimiert zu transferieren. Dieser Punkt untersucht die Möglichkeiten, die ein System bietet, um den Datentransfer zu optimieren. Dabei soll auch untersucht werden, ob für den Transfer der Daten effiziente Transport-Protokolle (auf der Ebene von HTTP oder SSH) genutzt werden können.

3.4.4 Potentielle Optimierungsmöglichkeiten der Datenbereitstellung

Hier wird untersucht, inwieweit Datenverarbeitungsschritte durch die Systeme optimiert werden. Eine Möglichkeit zur Optimierung ist die Restrukturierung von Workflow-Modellen, die entsprechende Datenquellenabfragen und Datenverarbeitungsschritte enthalten, zur Erreichung bestimmter Optimierungsziele. Die automatische Zusammenfassung mehrerer Datenzugriffe zu einem Datenzugriff während der Modellierungszeit ist z.B. ein solches Optimierungsziel. Dies wird in [VSS+07] näher beschrieben. Die Flexibilitätsaspekte aus den Kapiteln 3.3.5 und 3.3.6 können auch zur Optimierung genutzt werden. Dies könnte beispielsweise so aussehen, dass für bestimmte Datenmengen, die richtige Datenquelle zur Laufzeit mithilfe definierter Kriterien gefunden wird, in der die Daten bei einem Datentransfer gespeichert werden können. Mögliche Kriterien können dann sein, dass die Datenquelle überhaupt genug Speicherplatz aufweist oder dass die nachgelagerten Operationen (DM-Operationen oder Berechnungen) auf dieser Datenquelle bzw. einer dahinterliegenden Ressource (z.B. dem Rechner) effizient durchgeführt werden können. Das in Kapitel 3.3.8 erwähnte implizite und explizite Wissen über auszuführende Prozesse (enthaltene DM-Operationen und benötigte Datenquellen) kann ebenfalls für die Optimierung hilfreich sein [RRS+11].

3.5 Anbindung und Integration

In diesem Abschnitt wird auf Kriterien im Zusammenhang mit der Anbindung und Integration der Systeme eingegangen.

3.5.1 Integration von Werkzeugunterstützung

Die Werkzeugunterstützung ist ein entscheidendes Kriterium, da sie die Bedienbarkeit eines Systems erleichtert und einen einfacheren Zugang dazu liefert. Wichtig hierbei ist auch die Art der Werkzeugunterstützung, d.h. ob verschiedene Module bzw. Funktionalitäten in einem integrierten Werkzeug zur Verfügung gestellt werden oder ob es eine Vielzahl von Werkzeugen gibt, die dem Nutzer jeweils nur Teile des Systems zugänglich machen.

3.5.2 Anbindungsmöglichkeiten an andere Systeme/Dienste

In diesem Punkt wird geprüft, welche Anbindungsmöglichkeiten die untersuchten Datenbereitstellungssysteme anderen Systemen oder Diensten zur Nutzung der Datenbereitstellungssysteme bieten. Die Anbindung kann dabei beispielsweise über ein vom Datenbereitstellungssystem mitgelieferten Application Programming Interface (API) und einen entsprechenden Quellcode zur Einbindung dessen in das System bzw. den Dienst erfolgen. Die Anbindung kann auch über einen Client zur Verwendung einer bereitgestellten Web Service Schnittstelle erfolgen.

3.5.3 Fähigkeit verschiedene wissenschaftliche Programme zu koppeln

Software zur Durchführung von wissenschaftlichen Simulationen ist meist benutzerspezifisch oder problemspezifisch implementiert und wenig standardisiert. Beinahe jede Simulationssoftware verwendet daher eigene Datenformate für die Ein- und Ausgabedaten oder Zwischenergebnisse einer Simulation. Demgegenüber steht das Interesse verschiedene Simulationen und damit auch die entsprechende Software zu koppeln. Die so bereitgestellten gekoppelten Simulationen erlauben es Simulationen auf mehreren Skalen (multi-scale), in mehreren Domänen (multi-domain) und verschiedenen Bereichen der Physik (multi-physics) durchzuführen. Damit dies funktionieren kann, benötigt man ein entsprechendes Datenbereitstellungssystem, das die Datenintegration zwischen den verschiedenen Datenformaten realisiert, damit die Ausgabedaten der einen Simulation als Eingabedaten einer anderen Simulation verwendet werden können. In diesem Punkt werden dafür die Ergebnisse bestimmter Kriterien (siehe z.B. Kapitel 3.2.2, 3.3.1, 3.3.2 und 3.3.3) noch einmal zusammengefasst, die bestimmen, inwieweit sich die untersuchten Systeme für eine solche Kopplung eignen.

4 Evaluation von OGSA-DAI

In diesem Kapitel wird OGSA-DAI anhand der in Kapitel 3 definierten Kriterien untersucht.

4.1 Allgemeines

Dieser Abschnitt enthält die Evaluation der Kriterien, mit denen allgemeine Anforderungen an die Systeme untersucht wurden.

4.1.1 Anwendungsbereich

Das primäre Ziel bei der Entwicklung von OGSA-DAI ist die Bereitstellung einer einheitlichen Schnittstelle für das Datenmanagement und die Datenintegration in Anwendungen. Weiterhin soll durch OGSA-DAI der Austausch von heterogenen Daten zur Informationsgewinnung vereinfacht und vor allem der Zugriff auf und die Verarbeitung von verteilten Daten, d.h. Daten aus verschiedenen Datenquellen, erleichtert werden. OGSA-DAI bietet dazu entsprechende Webservice-Schnittstellen, die die Anbindung an Anwendungen in verschiedensten Umgebungen ermöglichen soll. So kann OGSA-DAI beispielsweise problemlos für die Datenintegration und das Datenmanagement in Grid- und Cloud-Umgebungen eingesetzt werden.

4.1.2 Bedienbarkeit, Einfachheit

Für OGSA-DAI wird momentan noch kein umfassendes Graphical User Interface (GUI) bereitgestellt. Es existieren lediglich einige Beispiel-Clients, die auch zum Teil als Java-Applet eine GUI besitzen. Der Funktionsumfang ist dabei allerdings sehr beschränkt und wirklich nur für Testzwecke ausreichend. Für die Modellierung der OGSA-DAI Workflows wird ebenfalls noch keine GUI bereitgestellt. Diese müssen von Hand in einem Texteditor erstellt werden. Gerade bei hochgradig verschachtelten Workflows kann dies unter Umständen sehr kompliziert und unübersichtlich werden. Einige Informationen über die bereitgestellten Aktivitäten und Ressourcen eines OGSA-DAI-Servers können über den Aufruf entsprechender Webseiten angezeigt werden. Dies ist vor allem für die Modellierung von Hand hilfreich, da man so eine Liste der nutzbaren Aktivitäten, Ressourcen und Datenquellen erhält. Die Konfiguration und Administration des Servers verläuft ebenfalls vollständig ohne

GUI durch Konfigurationsdateien und Kommandozeilenaufrufe. Dies wird im nächsten Abschnitt ausführlicher beschrieben.

4.1.3 Installation und Administration

OGSA-DAI wird in zwei Versionen bereitgestellt, zum einen mit Apache Axis [AXI] und zum anderen mit dem Globus Toolkit (GT) [GLO]. Die Axis-Distribution ist im Grunde die Standardversion von OGSA-DAI. Die GT-Distribution bringt durch das Globus Toolkit noch einige Grid-Funktionalitäten mit sich, wie beispielsweise die Komponente GridFTP. Diese stellt einen schnellen, sicheren und zuverlässigen FTP-Server bereit, der über einen Web Services Resource Framework (WSRF)-konformen Webservice verwendet werden kann. Beide Distributionen müssen zur Verwendung in einem Apache Tomcat [TOM] Server deployed werden. Die Installation muss weitgehend manuell, meist über die Kommandozeile, ausgeführt werden. Dank einer guten und detaillierten Installationsanleitung stellt das aber kein größeres Problem dar. Bevor OGSA-DAI allerdings installiert werden kann, müssen eine Java Runtime Environment (JRE), Apache Tomcat und Apache Ant installiert worden sein. Bei der Installation des Apache Tomcat Servers ist darauf zu achten, dass keine Leerzeichen im Installationspfad enthalten sind, da sonst OGSA-DAI nicht automatisch installiert werden kann. Apache Ant [ANTb] ist ein Kommandozeilenwerkzeug, mithilfe dessen in sogenannten Build-Dateien beschriebene Prozesse ausgeführt werden können. Es wird vornehmlich beim Compilieren und Erstellen von Java-Applikationen verwendet, kann aber im Grunde für die Ausführung von beliebig definierten Prozessen verwendet werden. Im Fall von OGSA-DAI wird die Installation und Administration zu großen Teilen mithilfe von Apache Ant realisiert. Sobald die benötigte Software installiert und alle entsprechenden Umgebungsvariablen gesetzt wurden, kann mittels `ant -Dtomcat.dir=%CATALINA_HOME% deploy` OGSA-DAI auf dem vorhandenen Tomcat-Server automatisch deployed werden. Damit ist die Installation von OGSA-DAI auch schon abgeschlossen und der Tomcat-Server kann gestartet werden.

Die Administration von OGSA-DAI erfolgt über die manuelle Anpassung von Einstellungsdateien im OGSA-DAI-Verzeichnis des Tomcat-Servers (`//Tomcat7/webapps/dai/`) sowie über die Erstellung von sogenannten Konfigurationsdateien und deren Verarbeitung mittels Apache Ant. Listing 4.1 zeigt eine Konfigurationsdatei, mit der eine neue Data Resource erstellt und konfiguriert wird. Dabei wird in Zeile 1 definiert, dass eine neue *Java Database Connectivity (JDBC)*-Ressource über den Befehl *deployMySQL* mit dem Namen *MySQLResource* und der URL *jdbc:mysql://myhost:3306/daitest* erstellt werden soll. In Zeile 2 wird anschließend noch der Zugriff auf die in Zeile 1 erstellte Ressource definiert. Dabei wird in diesem Fall eine einfache Authentifizierung mittels Benutzername und Passwort festgelegt. D.h. wenn später diese Ressource verwendet wird, werden automatisch die in Zeile 2 hinterlegten Informationen zur Authentifizierung für jeden Client genutzt. Möchte man dies nicht, kann man den Platzhalter *ANY* durch die Angabe entsprechender Attribute oder eindeutiger Namen ersetzen. Diese werden dann zur Authentifizierung des Clients gegenüber OGSA-DAI verwendet und anschließend zur Authentifizierung auf der Datenquelle durch die hinterlegten Benutzernamen und Passwörter ersetzt. Hat man nun die Konfigurationsdatei erstellt, muss diese mittels Apache Ant ausgeführt werden. Listing 4.2 zeigt das Schema

Listing 4.1 Beispiel einer Konfigurationsdatei aus [CEMP11]

```
1 JDBC deployMySQL MySQLResource jdbc:mysql://myhost:3306/daitest
2 Login permit MySQLResource ANY myUser myPassword
```

Listing 4.2 Schema des Konfigurationsaufrufs mit Apache Ant aus [CEMP11]

```
ant -Dtomcat.dir=%CATALINA_HOME% -Dconfig.file=CONFIG-FILE [-Djar.dir=JAR-DIRECTORY]
    [-Dstart.line=LINE] configure
```

des entsprechenden Kommandozeilenbefehls zur Verarbeitung der Konfigurationsdateien. Der Inhalt der Konfigurationsdatei hat dabei keinen Einfluss auf den Befehl. Dieser gliedert sich in folgende Teile: Mittels `ant` wird Apache Ant gestartet und die nachfolgenden Angaben als Parameter zur Abarbeitung der hinterlegten Build-Datei gekennzeichnet. Über `-Dtomcat.dir=%CATALINA_HOME%` kann das Tomcat-Verzeichnis angegeben werden. Da eine entsprechende Umgebungsvariable gesetzt wurde, kann der Pfad über diese Umgebungsvariable (`%CATALINA_HOME%`) angegeben werden. Mittels `-Dconfig.file=CONFIG-FILE` muss der relative Pfad zur Konfigurationsdatei sowie deren Namen angegeben werden. Optional kann über `-Djar.dir=JAR-DIRECTORY` ein Ordner mit benötigten Java-Archiven (JARs) angegeben werden. Dies wird beispielsweise bei der Registrierung von selbst-implementierten Aktivitäten oder Ressourcen benötigt. Durch Angabe des optionalen Parameters `-Dstart.line=LINE` kann noch eine Zeilennummer angegeben werden, ab welcher die Konfigurationsdatei verarbeitet werden soll. Dies ist hilfreich, falls Angaben am Anfang der Datei ignoriert werden sollen oder in einer Datei mehrere Konfigurationsschritte enthalten sind, die nacheinander über mehrere Ant-Aufrufe abgearbeitet werden sollen. Zu guter Letzt wird durch die Angabe von `configure` der anzustoßende Prozess ausgewählt. Bei der Installation wurde so analog dazu mit einem ähnlichen Befehl der deploy-Prozess angestoßen. Auf diese Art und Weise lassen sich alle vorhandenen Aktivitäten und Ressourcen konfigurieren sowie neue Aktivitäten und Ressourcen bereitstellen. Dabei sind die Änderungen des OGSA-DAI-Servers sofort aktiv und erfordern keinen Neustart des Servers. Dagegen muss nach der Änderung von Einstellungsdateien immer ein Neustart des Servers durchgeführt werden, da die Einstellungsdateien nur während des Startvorgangs neu eingelesen werden.

4.1.4 Abhängigkeiten von anderer Software

OGSA-DAI benötigt mindestens Apache Tomcat Version 5.5 oder höher, JRE Version 1.6 oder höher sowie Apache Ant Version 1.6 oder höher. Zur graphischen Visualisierung der Ausführung von Workflows in einem bereitgestellten Beispiel-Client wird Graphviz [GRA] benötigt. Diese Abhängigkeiten haben keinerlei Implikationen auf die Verwendung von OGSA-DAI, da die abhängigen Tools alle plattformunabhängig sind und OGSA-DAI selbst über eine abstrakte Webservice-Schnittstelle genutzt wird, die die entsprechende Integration im Hintergrund kapselt. Lediglich die Verwendung des Globus Toolkits bzw. der entsprechenden OGSA-DAI Distribution schränkt die Verwendung von OGSA-DAI auf Unix-Plattformen ein, da das Globus Toolkit nur auf diesen Plattformen verwendet werden kann.

4.1.5 Dokumentation

Die Dokumentation ist sehr gut strukturiert und enthält detaillierte Informationen zur Installation, Administration, Nutzung und Erweiterung von OGSA-DAI. Die grundlegende Struktur ist dabei so angelegt, dass der Leser zuerst eine Einführung in die Grundlagen von OGSA-DAI erhält. Anschließend folgt eine Installationsanleitung sowie detaillierte Anleitungen zur Nutzung und Erweiterung von OGSA-DAI. Diese sind jeweils sowohl aus Sicht der Nutzer als auch für Entwickler vorhanden. So können sich Nutzer über die Verwendung von OGSA-DAI und den zugehörigen Clients informieren, und auch Entwickler erhalten so benötigte Informationen zur Erweiterung von OGSA-DAI und zur Entwicklung von Clients. Weiterhin finden sich auf der Homepage von OGSA-DAI (http://sourceforge.net/apps/trac/ogsa-dai/wiki) eine Vielzahl weiterer Hilfestellungen wie FAQs, Mailinglisten oder bekannte Bugs und Fixes.

4.2 Softwarequalität

Dieser Abschnitt enthält die Evaluationsergebnisse einiger Kriterien, die sich mit der Qualität der Softwaresysteme befassen.

4.2.1 Portabilität/Plattformunabhängigkeit

OGSA-DAI selbst ist plattformunabhängig, da es vollständig in Java realisiert ist und Java die Plattformunabhängigkeit gewährleistet. Apache Tomcat und Apache Ant sind ebenfalls plattformunabhängig und schränken so die Portabilität von OGSA-DAI nicht ein. Lediglich das Globus Toolkit ist an Unix-Plattformen gebunden und kann daher zusammen mit OGSA-DAI nur auf Unix-Systemen eingesetzt werden.

4.2.2 Erweiterbarkeit

OGSA-DAI stellt eine Menge von Erweiterungsmöglichkeiten über die Implementierung definierter Schnittstellen bereit. So können z.B. neue Aktivitäten, Data Resources, Security-Funktionalität oder auch eine komplett neue Präsentationsschicht implementiert und in OGSA-DAI angebunden werden. Die Dokumentation liefert dazu detaillierte Anleitungen mit einigen Beispielen und Tipps. Nachfolgend wollen wir uns die Erweiterung von OGSA-DAI am Beispiel einer selbst implementierten MySQL Data Resource exemplarisch ansehen. Um eine neue Data Resource zu erstellen, muss das Interface *uk.org.ogsadai.resource.dataresource.DataResource* implementiert werden. Dieses Interfaces definiert die zu implementierende Funktionalität einer Data Resource wie beispielsweise das Öffnen und Schließen von Verbindungen zur physischen Datenquelle oder die Bereitstellung von Metadaten einer Datenquelle für Aktivitäten. Eine detaillierte Beschreibung liefert [CEMP11] (Part IV, Chapter 111). Sobald die Implementierung abgeschlossen ist, muss

Listing 4.3 Beispiel einer Data Resource Konfigurationsdatei aus [CEMP11]

```
id=MyResource
type=uk.org.ogsadai.DATA_RESOURCE
creationTime=null
terminationTime=null
PROPERTIES
END
CONFIG
  dai.driver.class=org.gjt.mm.mysql.Driver
  dai.data.resource.uri=jdbc:mysql://myhost:3306/daitest
  dai.login.provider=uk.org.ogsadai.LOGIN_PROVIDER
END
ACTIVITIES
  uk.org.ogsadai.SQLQuery=uk.org.ogsadai.SQLQuery
END
dataResourceClass=org.mydomain.MyDataResource
```

für die neue Data Resource eine Konfigurationsdatei erstellt werden. Diese wird später auf den OGSA-DAI Server deployed und wird zur Laufzeit zur Instanziierung von entsprechenden Data Resource Objekten genutzt. Listing 4.3 zeigt ein Beispiel einer solchen Ressourcen-Konfigurationsdatei. Dabei ist wichtig, dass die angegebene Ressourcen-ID und der Dateiname der Konfigurationsdatei identisch sind. Ebenfalls wichtig ist die Angabe des Ressourcen-Typs. Dabei stehen die in Kapitel 2.1.1 beschriebenen sechs Ressourcen-Typen zur Verfügung. Weiterhin muss für relationale Datenquellen eine JDBC-Treiberklasse, die Datenbank-URL und die zu nutzende Login-Provider-Implementierung angegeben werden. Zu guter Letzt kann noch eine Liste mit Aktivitäten angegeben werden, die mit dieser Datenquelle als Ressource ausgeführt werden können. Diese Ressourcen-Konfigurationsdateien haben dabei immer dasselbe Schema und erlauben so die detaillierte Konfiguration jeglicher Ressourcen, auch solcher, die nicht selbst implementiert wurden. Die in Listing 4.1 in Kapitel 4.1.3 aufgezeigte Konfigurationsdatei enthält eine Reihe abstrakterer Angaben, die dem Nutzer das Deployment neuer Standard-OGSA-DAI Ressourcen vereinfachen. Im Hintergrund wird aus diesen Angaben allerdings wieder eine Ressourcen-Konfigurationsdatei, nach dem in Listing 4.3 dargestellten Schema erstellt. Die ausführliche Konfiguration stellt dabei mehr Möglichkeiten bereit. So lassen sich beispielsweise auch Lese- und Schreibzugriffe einer Datenquelle einschränken. In Listing 4.3 wurde dazu nur eine lesende Aktivität (SQLQuery) angegeben, dadurch ist es später keinem Client möglich Daten der definierten Datenbank zu ändern oder Daten in diese einzufügen.

Die erstellte Data Resource Implementierung kann nun mithilfe der Ressourcen-Konfigurationsdatei (MyResource.txt) in OGSA-DAI deployed werden. Dazu muss eine weitere Konfigurationsdatei (deployResource.txt) erstellt werden, die eine entsprechende Zeile (`Resource deployResource MyNewResource MyResource.txt`) enthält, um die neue Data Resource zusammen mit ihrer Konfigurationsdatei (MyResource.txt) auf dem Server zu deployen. Anschließend muss diese zweite Konfigurationsdatei (deployResource.txt), wie bereits in Kapitel 4.1.3 aufgezeigt, mittels Apache Ant ausgeführt werden. Hierbei ist darauf zu achten, dass im Ant-Befehl der Ordner angegeben wird, in dem die JAR mit der Data Resour-

ce Implementierung liegt (`ant -Dtomcat.dir=%CATALINA_HOME% -Dconfig.file=deployResource.txt -Djar.dir=JAR-DIRECTORY configure`).

4.3 Funktionsumfang

In diesem Abschnitt wird auf die Evaluationsergebnisse der Kriterien im Zusammenhang mit dem Funktionsumfang von OGSA-DAI eingegangen.

4.3.1 Datenquellen

Grundlegend können durch die Erweiterbarkeit von OGSA-DAI (siehe Kapitel 4.2.2) alle Datenquellen angebunden werden, die eine entsprechende Java-API bereitstellen oder durch eine solche erreichbar sind. Im Moment werden bereits standardmäßig folgende Datenquellen unterstützt:

relationale Datenbanken MySQL, PostgreSQL, Oracle, Microsoft SQLServer und IBM DB2

XML-Datenbanken eXist

Dateisysteme Alle Dateisysteme, die über Java zugänglich sind

Remote-Datenquellen Datenquellen, die an andere OGSA-DAI-Server angebunden sind

Resource Description Framework (RDF)-Datenquellen JenaDB und SPARQL-Endpunkte

Sonstige Über entsprechende Aktivitäten können weitere Datenquellen genutzt werden. So können z.B. Dateien von FTP-Servern abgerufen oder auf diese übertragen werden. Ebenfalls möglich ist der Abruf von Dateien über eine URL mittels HTTP oder auch das Versenden von Daten per E-Mail über einen SMTP-Server.

Dabei können Datenquellen, für die es einen entsprechenden JDBC-Treiber gibt, problemlos ohne Implementierung über Konfigurationsdateien in OGSA-DAI eingebunden werden (siehe Kapitel 4.1.3 und 4.2.2). Zur Authentifizierung auf Datenquellenseite werden die in der Data Resource bzw. einer Aktivität hinterlegten Authentifizierungsinformationen (Benutzername und Passwort) genutzt.

4.3.2 Datenformate

OGSA-DAI nutzt die Java Basis-Datentypen sowie einige OGSA-DAI-spezifische Datentypen zur Verarbeitung von Daten und Datenquellen-Metadaten. Listing 4.4 zeigt die von OGSA-DAI unterstützten Datentypen. Die Datentypen `uk.org.ogsadai.activity.io.ControlBlock.LIST_BEGIN` und `uk.org.ogsadai.activity.io.ControlBlock.LIST_END` werden dabei zur Identifizierung des Anfangs und Endes von Listen verwendet. Diese sind notwendig, damit zur Laufzeit

Listing 4.4 Von OGSA-DAI unterstützte Datenquellen aus [CEMP11]

```
java.lang.String
char[]
byte[]
java.sql.Blob
java.sql.Clob
java.lang.Boolean
java.lang.Float
java.lang.Integer
java.lang.Long
java.lang.Double
java.lang.Date
java.lang.Calendar
uk.org.ogsadai.activity.io.ControlBlock.LIST_BEGIN
uk.org.ogsadai.activity.io.ControlBlock.LIST_END
uk.org.ogsadai.tuple.Tuple
uk.org.ogsadai.tuple.TupleMetadata
uk.org.ogsadai.tuple.MetadataWrapper
org.w3c.dom.Node
```

auch bei der parallelen Verarbeitung von mehrfach verschachtelten Listen immer klar ist, wann eine Liste fertig abgearbeitet wurde und die nächste Liste beginnt. Die Datentypen *Tuple* und *TupleMetadata* vereinfachen die Verarbeitung und Bereitstellung von relationalen Daten (Tuple) und deren Metadaten (TupleMetadata). Der Datentyp *MetadataWrapper* ermöglicht die Kapselung von Objekten, die als Metadaten angesehen werden sollen (wie beispielsweise TupleMetadata). So können anwendungsspezifische Metadaten innerhalb von OGSA-DAI bereitgestellt und verarbeitet werden. XML-Daten werden durch die Klasse org.w3c.dom.Node repräsentiert. Da die internen Datentypen von OGSA-DAI sich stark an denen von Java orientieren oder sogar meist die identischen Datentypen genutzt werden, muss meistens keine Transformation der Daten stattfinden. Eine Ausnahme bildet das Tuple-Datenformat. Dieses wird nur intern von OGSA-DAI verwendet, während JDBC ein ResultSet-Datentyp für das Ergebnis einer Datenquellenanfrage nutzt. Die Konvertierung zwischen Tuple- und ResultSet-Datentyp übernimmt OGSA-DAI allerdings automatisch in beide Richtungen. Der Nutzer hat aber auch die Möglichkeit über die Nutzung entsprechender Aktivitäten Daten explizit in andere Formate zu transformieren. So ist beispielsweise mit der *TupleToWebRowSetCharArrays*-Aktivität die Konvertierung von Daten im Tuple-Datenformat in eine entsprechende XML-Repräsentation möglich. Eine Übersicht der verfügbaren Aktivitäten liefert die *Activities Reference* aus [CEMP11] (URL: http://ogsa-dai.sourceforge.net/documentation/ogsadai4.1/ogsadai4.1-axis/ActivitiesReference.html).

4.3.3 Datenzugriffstechnologien

OGSA-DAI nutzt JDBC als Datenzugriffstechnologie für alle relationalen und XML-Datenbanken. Der Zugriff auf Dateisysteme wird über die JavaIO API realisiert. Über

Ebene	Pattern	OGSA-DAI-Aktivitäten
ETL Patterns / ETL Operations	Selektion	SQLQuery, SQLNestedInClauseQuery, SQLParameterisedQuery, TupleSelect, ReadFromFile, XPathQuery, XQuery, EPRQuery und RDFDBQuery
	Projektion	TupleProjection und TupleProjectByIDS
	Verbund	GenericTupleJoinActivity, SQLNestedInClauseJoin, TupleJoin, TupleMergeJoin und TupleSemiJoin
	Kreuzprodukt	TupleProduct, ListMultiply
Basic Data Management Patterns	Data Split	Split, RandomSplit, ListRandomSplit und TupleSplit
	Data Merge	TupleSimpleMerge, ListConcatenate, TupleUnionAll
	Data Iteration	alle Aktivitäten iterieren automatisch über eingehende Daten

Tabelle 4.1: Datenmanagement-Patterns und OGSA-DAI Aktivitäten

Webservices können lediglich die Datenquellen entfernter OGSA-DAI-Server abgefragt werden. Dazu werden die von einem entfernten OGSA-DAI-Server verwalteten Datenquellen als sogenannte *Remote Resources* im lokalen OGSA-DAI-Server deployed. Durch Angabe der URL des enternten OGSA-DAI-Servers und der dort vergebenen Ressourcen-ID kann so zur Laufzeit die entsprechende Datenquellen-Anfrage an den entfernten OGSA-DAI-Server (Webservice-Schnittstelle) weitergeleitet, dort verarbeitet und das Ergebnis zurückgeliefert werden.

4.3.4 Datenverarbeitungsmöglichkeiten / Datenmanagement-Patterns

Durch entsprechende Aktivitäten werden einige der Datenmanagement-Patterns der Ebenen „ETL Patterns / ETL Operations" und „Basic Data Management Patterns" unterstützt. Natürlich können durch die Modellierung und Ausführung entsprechender (Sub-)Workflows mithilfe dieser Aktivitäten auch Patterns höherer Ebenen realisiert werden. Dies kann ebenso durch die Implementierung weiterer Aktivitäten erreicht werden. Tabelle 4.1 gibt einen Überblick über die jeweiligen Patterns und möglichen Aktivitäten, die diese Patterns unterstützen. Einige Aktivitäten unterliegen dabei gewissen Einschränkungen. Beispielsweise kann nicht angegeben werden, wie ein Datenstrom durch eine Split-Aktivität aufgeteilt werden soll, sondern der Datenstrom wird automatisch über ein Round-Robin-Verfahren aufgeteilt. Auf die Einschränkungen der einzelnen Aktivitäten wird allerdings im Rahmen dieses Dokuments nicht näher eingegangen. Detaillierte Informationen dazu liefert die *Activities Reference* aus [CEMP11].

4.3.5 Flexibilität zur Deploymentzeit

OGSA-DAI bietet derzeit keine Möglichkeit auf den in Kapitel 2.1.2 beschriebenen Deployment-Prozess (*Einlesen der Workflow-Datei, Instanziierung aller modellierten Aktivitä-*

ten, Verknüpfung der Aktivitäten mit den referenzierten Ressourcen) Einfluss zu nehmen. Alle Aktivitäten und Ressourcen werden statisch über ihre ID im Workflow referenziert und von OGSA-DAI zur Instanziierung identifiziert. Durch entsprechende Anpassungen könnte man den Deployment-Mechanismus beispielsweise so erweitern, dass einem Workflow erst zur Deploymentzeit konkrete Ressourcen zugeordnet werden. So ließe sich eine Wiederverwendung von Workflow-Modellen in verschiedenen OGSA-DAI Umgebungen realisieren, da ein Workflow nicht mehr direkt an konkrete Ressource-IDs gebunden wäre.

4.3.6 Flexibilität zur Laufzeit

Konzepte für die Flexibilität zur Laufzeit werden momentan ebenfalls noch nicht unterstützt. Allerdings wäre auch hier eine entsprechende Erweiterung möglich. Beispielsweise könnte eine Late-Binding Funktionalität implementiert werden, die anhand von im Workflow angegebenen Kriterien zur Laufzeit automatisch eine passende Datenquelle auswählt und anbindet. Eventuell lässt sich dies vielleicht sogar über die Implementierung einer entsprechenden Data Resource realisieren, die bei ihrer Instanziierung eine passende Datenquelle ausfindig macht und dann eine entsprechende Verbindung zu dieser Datenquelle bereitstellt.

4.3.7 Möglichkeit Anforderungen an Datenqualität zu formulieren

OGSA-DAI bietet derzeit keine Möglichkeit Anforderungen an Datenqualität zu formulieren. Allerdings könnte diese Funktionalität durch die Implementierung entsprechender Aktivitäten bereitgestellt werden. Diese Aktivitäten könnten dann in Workflows genutzt werden, um die Qualität der Daten anhand von definierten Kriterien während der Laufzeit zu evaluieren.

4.3.8 Transparenz der Datenbereitstellung und des Datenmanagements

Durch entsprechende Konfiguration des OGSA-DAI-Servers kann die Ausführung von Workflows aufgezeichnet werden. Über eine Java Server Page (JSP) (dai-trace.jsp) kann anschließend die Aufzeichnung der Ausführung aktiviert bzw. deaktiviert werden. Die Aufzeichnungen werden dabei in Dateien in einem Ordner auf dem Server gespeichert. Die so aufgezeichneten Daten werden über dieselbe JSP visualisiert. Zur Visualisierung des Workflow-Graphen wird dabei Graphviz [GRA] benötigt. Dieser Mechanismus ist allerdings eher für den Test und die Fehlerbehebung von komplexen Workflows gedacht und nicht für das Monitoring der Ausführung. Dafür stellt OGSA-DAI eine entsprechende Schnittstelle (`uk.org.ogsadai.monitoring.MonitoringFramework`) bereit. Durch die Implementierung dieser Schnittstelle können EventListener registriert werden, die von Aktivitäten, Pipelines und Ressourcen emittierte Events sammeln und für das Auditing, die Provenance oder das Monitoring eines Workflows bereitstellen. OGSA-DAI liefert hierfür bereits eine Beispiel-Implementierung (`uk.org.ogsadai.monitoring.example.EventListMonitoringFramework`), die einfach alle emittierten Events sammelt und in einer Liste im Arbeitsspeicher hält.

4.4 Leistungsfähigkeit

Dieser Abschnitt beschreibt die Evaluationsergebnisse bzgl. der Kriterien, die sich mit der Leistungsfähigkeit der Systeme befassen. In den nachfolgenden Punkten wird dabei nicht mehr auf die in Kapitel 2.1.1 bereits beschriebene Optimierungsmöglichkeit des pipeline-artigen Datenflusses eingegangen, sondern weitere Konzepte zur Erhöhung der Leistungsfähigkeit von OGSA-DAI aufgezeigt.

4.4.1 Performanz des Datenzugriffs

OGSA-DAI unterstützt die Verwendung von Datenbankverbindungspools (Connection-Pools) für relationale Datenbanken. Dabei wird mithilfe von Apache Commons DBCP [DBC] für jede registrierte relationale Ressource ein konfigurierbarer Connection-Pool bereitgestellt.

4.4.2 Performanz der Datenverarbeitung

Mithilfe von Distributed Query Processing (DQP) ermöglicht OGSA-DAI die Ausführung von Abfragen (Queries) auf Tabellen mehrerer verteilter relationaler Datenbanken über SQL, so als ob die Tabellen in einer gemeinsamen Datenbank liegen würden. Dazu wird eine sogenannte *DQP Resource* erstellt, die zur Laufzeit eine virtuelle Datenbank bereitstellt, deren Schema aus der Vereinigung der Schemen aller referenzierten verteilten Datenbanken generiert wird. Dadurch sind über die generierte virtuelle Datenbank alle Tabellen der referenzierten Datenbanken verfügbar und es können SQL-Abfragen an die virtuelle Datenbank gestellt werden. Die Tabellen erhalten dabei als Präfix die Ressourcen-ID der relationalen Ressource, in der sie liegen. Der entscheidende Vorteil von DQP liegt darin, dass keinerlei Daten lokal zwischengespeichert werden, sondern die Abfragen direkt auf den referenzierten relationalen Datenbanken ausgeführt und nur die Abfrageergebnisse zurückgeliefert werden. Abbildung 4.1 zeigt ein Beispiel-Szenario, in dem eine SQL-Abfrage auf einer DQP Resource ausgeführt wird. MyDQPResource referenziert dabei die beiden relationalen Ressourcen MySQLResource und PostgreSQLResource. Diese enthalten jeweils eine Tabelle, die in der Abfrage durch die Ressourcen-ID und den Tabellennamen (z.B. MySQLResource_TabelleA) angesprochen werden kann. Die relationalen Ressourcen können dabei auch auf mehreren verschiedenen OGSA-DAI-Servern liegen.

OGSA-DAI bietet weiterhin die Möglichkeit Aktivitäten auf ganze Gruppen von Ressourcen (Resource Group) auszuführen. So kann beispielsweise eine SQL-Abfrage auf mehreren relationalen Datenbanken mithilfe der SQLBag-Aktivität ausgeführt werden. Dies wird realisiert, indem ein Subworkflow, der die SQL-Abfrage auf einer einzigen relationalen Ressource ausführt, für jede in der Resource Group enthaltene Ressource generiert und ausgeführt wird. Alle Subworkflows laufen dabei wieder parallel ab, so dass auch hier eine hohe Performanz gewährleistet wird. Das Ergebnis aller Subworkflows wird dann als Liste von Tuple-Daten (Ergebnis eines Subworkflows) bereitgestellt.

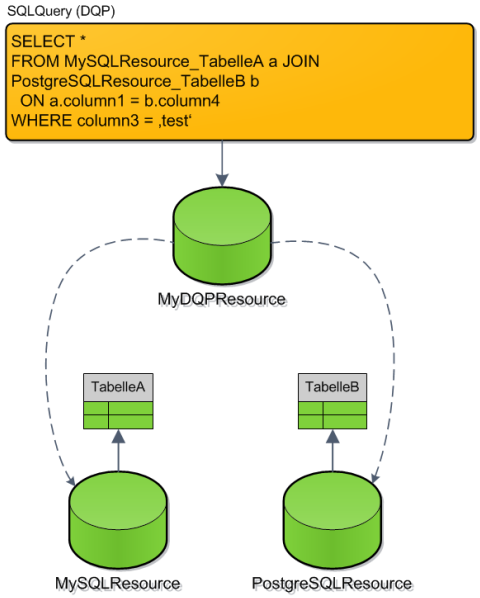

Abbildung 4.1: SQLQuery auf mehreren verteilten relationalen Datenbanken mit DQP (vgl. [CEMP11])

Für Dateisysteme bietet OGSA-DAI die Möglichkeit an Dateien bzw. deren Inhalt zu indizieren, um diese effizienter durchsuchen zu können. Zur Bereitstellung dieser Funktionalität nutzt OGSA-DAI Apache Lucene [LUC], eine java-basierte Bibliothek für die Indizierung und Volltextsuche.

4.4.3 Performanz des Datentransfers

Die Nutzung von expliziten Referenzen auf Daten ist in OGSA-DAI nicht möglich. Es wird allerdings ein referenz-ähnlicher Mechanismus bereitgestellt. OGSA-DAI erlaubt es die Ergebnisse einer Abfrage asynchron zu übertragen. D.h. die Ergebnis-Daten der Abfrage werden nicht direkt an den Client zurückgeliefert, sondern auf dem OGSA-DAI-Server in einer neu erstellten Data Resource gehalten und dem Client lediglich die ID dieser Ressource übermittelt. Dadurch kann der Client selbst irgendwann die Daten vom OGSA-DAI-Server abfragen oder auch die Ressourcen-ID zusammen mit der OGSA-DAI-Server URL an eine andere Anwendung weiterleiten. Die Ressourcen-ID kann daher als eine Referenz auf die Daten angesehen werden und direkt von der Anwendung aufgelöst werden, die die Daten tatsächlich benötigt.

4.4.4 Potentielle Optimierungsmöglichkeiten der Datenbereitstellung

OGSA-DAI bietet im Moment keine integrierten Optimierungsmöglichkeiten an. Allerdings besteht die Möglichkeit eingehende Workflows vor ihrer Ausführung auf dem Server zu transformieren. Dazu können benutzerspezifische Transformatoren implementiert und registriert werden. Über diese könnten dann entsprechende Anpassungen oder auch Optimierungen des Workflows durchgeführt werden.

4.5 Anbindung und Integration

In diesem Abschnitt wird auf die Evaluationsergebnisse bzgl. der Kriterien im Zusammenhang mit der Anbindung und Integration der Systeme eingegangen.

4.5.1 Integration von Werkzeugunterstützung

Der Grundgedanke von OGSA-DAI ist die Bereitstellung einer standardisierten Schicht zwischen Anwendungen und Datenquellen. Anwendungen können dadurch die Datenintegration und das Datenmanagement über OGSA-DAI realisieren, indem sie entsprechende Ressourcen definieren und Workflows zur Verarbeitung ihrer Daten bereitstellen. Daher existieren im Moment noch keine Werkzeuge die OGSA-DAI integrieren. Ein Werkzeug zur graphischen Modellierung und für den Test von OGSA-DAI Workflows wäre sicherlich trotzdem sinnvoll.

4.5.2 Anbindungsmöglichkeiten an andere Systeme/Dienste

OGSA-DAI bietet eine Webservice-Schnittstelle an, die über entsprechende Clients in anderen Systemen/Diensten eingebunden werden kann. Ebenso möglich ist die Anbindung von OGSA-DAI über eine bereitgestellte Representational State Transfer (REST)-Schnittstelle. Diese ermöglicht die Integration über HTTP und erlaubt so sogar die Anbindung von OGSA-DAI in Webseiten. Über diese beiden Schnittstellen kann OGSA-DAI in nahezu jede Anwendung völlig plattformunabhängig integriert werden.

4.5.3 Fähigkeit verschiedene wissenschaftliche Programme zu koppeln

Die effiziente Kopplung von (wissenschaftlichen) Anwendungen ist eines der Hauptziele von OGSA-DAI. Entscheidend dafür sind die Bereitstellung von standardisierten plattformunabhängigen Schnittstellen (siehe Kapitel 4.5.2), die Unterstützung einer Vielzahl von Datenformaten (siehe Kapitel 4.3.2) und Datenquellen (siehe Kapitel 4.3.1) sowie die Erweiterbarkeit (siehe Kapitel 4.2.2) von OGSA-DAI.

5 Evaluation von SDMCenter

Dieses Kapitel beschreibt das System SDMCenter nach den in Kapitel 3 festgelegten Kriterien. Da sich SDMCenter in mehrere Technologien und Systeme für die verschiedenen Ebenen des Datenmanagements aufgliedert, wurden für die Untersuchung nur die Technologien und Systeme betrachtet, die mit den anderen untersuchten Systemen vergleichbar sind und insbesondere für eine Verbesserung und Erweiterung von SIMPL relevant sind. Das einzige vergleichbare System von SDMCenter ist Kepler. Daher beziehen sich die folgenden Abschnitte ausschließlich auf Kepler und nicht auf SDMCenter als Gesamtsystem. Auf weitere Technologien und Systeme von SDMCenter, die für eine Verbesserung und Erweiterung von SIMPL relevant sind, wird ggf. bei der Beschreibung der Kriterien bzw. abschließend in Kapitel 6 eingegangen.

5.1 Allgemeines

Dieser Abschnitt enthält die Evaluation anhand der Kriterien, mit denen allgemeine Anforderungen an die Systeme untersucht wurden.

5.1.1 Anwendungsbereich

Die Zielgruppe von Kepler sind Wissenschaftler und Analysten, um diesen die Modellierung von wissenschaftlichen Workflows für Simulationen und Analysen zu ermöglichen. Dabei spezialisiert sich Kepler auf die Modellierung des Datenflusses und die Visualisierung von Simulations- und Analyse-Ergebnissen.

Mit Kepler ist es möglich auf Daten von verschiedenen Datenrepositories wie z.B. ökologische Daten von Feldstationen oder Daten von Geowissenschaftlern oder Museen zuzugreifen und diese zu verarbeiten. Dazu wird Kepler bereits mit einer Vielzahl von Aktoren für verschiedene Datenbereitstellungssysteme ausgeliefert (siehe Kapitel 5.3.1).

Kepler fördert außerdem den Austausch zwischen Wissenschaftlern, indem es Möglichkeiten bietet, erstellte Workflows über Online-Repositories auszutauschen.

5.1.2 Installation und Administration

Für die Installation existiert ein Installer, der den Benutzer durch den Installationsprozess führt und diesen dadurch sehr einfach gestaltet.

Die Administration von Kepler erfolgt über eine zentrale GUI (Graphical User Interface), die Zugriff auf alle Funktionen von Kepler bietet. Zusätzlich gibt es ein extra Management-Tool für die Verwaltung und Aktualisierung der Module von Kepler, über das auch die Installation weiterer öffentlich angebotener Module über das Internet möglich ist. Dadurch kann Kepler beispielsweise um eine Provenance-Funktionalität erweitert werden.

5.1.3 Bedienbarkeit, Einfachheit

Die GUI von Kepler ist sehr schlicht gehalten und bietet schnellen Zugriff auf wichtige Funktionen über eine konfigurierbare Toolbar und auf alle weiteren Funktionen über eine Menüleiste. Sie teilt sich in drei Bereiche auf, wobei es einen Bereich zur Auswahl der Elemente wie z.B. Aktoren und Direktoren gibt, einen Arbeitsbereich, in den die Elemente per Drag&Drop gezogen werden können, sowie eine Mini-View des Workflows, um auch bei größeren Workflows die Übersicht zu behalten. Technische Details zur Ausführung des Workflows werden vor dem Benutzer versteckt. Die Elemente bieten oft umfangreiche Möglichkeiten konfiguriert zu werden und es ist daher oft schwierig zwischen wichtigen und unwichtigen bzw. speziellen Einstellungen zu unterscheiden. Oft stehen auch interne Einstellungen der Elemente, die die Implementierung betreffen, mit zur Auswahl. Dadurch wird man selbst bei einfachen Vorhaben gezwungen sehr viel Dokumentation zu lesen. Für eine bessere Übersicht über die Ausführung gibt es ein extra Runtime-Fenster, in dem die Workflow-Steuerung, die Einstellungen der Direktoren und die Ausgabebereiche zusammengefasst werden. Außerdem lässt sich die Ausführung animiert darstellen, so dass gerade aktive Aktoren optisch hervorgehoben werden. Zur besseren Übersicht des Workflows gibt es einen Outline-Bereich, in dem alle Elemente des Workflows mit Ports, Attributen und Beziehungen in einer Baumstruktur dargestellt werden.

5.1.4 Abhängigkeiten von anderer Software

Kepler ist weitestgehend unabhängig von anderer Software und benötigt nur eine Laufzeitumgebung für Java (JRE - Java Runtime Environment). Die einzige Abhängigkeit besteht bei der statistischen Analyse, für die eine Installation von R benötigt wird, damit entsprechende Ausdrücke in R in Kepler ausgewertet werden können.

5.1.5 Dokumentation

Die Dokumentation von Kepler ist sehr umfangreich und bietet verschiedene PDF Dokumente. Neben einem User Guide und einer Actor Reference, die alle Aktoren erklärt, gibt es auch einen Getting Started Guide, der sich direkt an Wissenschaftler richtet und anhand von

mehreren Beispielen eine Schritt-für-Schritt-Anleitung für verschiedene Anwendungsfälle bietet. Dazu gibt es außerdem einige Beispiel-Workflows, die geladen und angepasst werden können, so dass für bestimmte Anwendungsfälle ein schneller Start möglich ist. Für das Ecological Niche Modeling (ENM) gibt es zudem einen extra Guide. Die Dokumentation ist auch aus dem Programm heraus erreichbar und steht zum Teil auch als Online-Hilfe zur Verfügung.

5.2 Softwarequalität

Dieser Abschnitt beschreibt einige Kriterien, die sich mit der Qualität der Softwaresysteme befassen.

5.2.1 Portabilität/Plattformunabhängigkeit

Da Kepler in Java geschrieben ist, kann es auf jedem Betriebssystem eingesetzt werden, für das eine JRE existiert. Dies ist für alle gängigen Betriebssysteme der Fall. Grundsätzlich kann Kepler auf andere Systeme portiert werden und ist dort lauffähig. Allerdings werden benutzerspezifische Daten in den jeweiligen Anwenderprofilen des Betriebssystems hinterlegt, die ebenfalls portiert werden müssen.

5.2.2 Erweiterbarkeit

Kepler ist ein Open Source Projekt und kann daher prinzipiell an alle Bedürfnisse angepasst werden. Zudem gibt es die Möglichkeit eigene Module und Aktoren für Kepler in Java zu schreiben, um die Funktionalität sowie den Umfang der Modellierungsmöglichkeiten zu erweitern. Dazu stehen die folgenden Erweiterungspunkte zur Verfügung.

Configuration System Kepler verfügt über eine zentrale Verwaltung aller Einstellungen des Systems. Über eine API kann diese von Aktoren und Modulen benutzt werden. Dadurch können z.B. die Einstelllungen eines Moduls modulübergreifend bereitgestellt werden und sind von beliebigen Stellen aus abrufbar und änderbar.

Menu System Über diesen Erweiterungspunkt können Menüs und Kontextmenüs erweitert werden.

Event-State Ermöglicht die Inter-Modulare Kommunikation über Events.

ViewPane und TabPane Über diese beiden Erweiterungspunkte kann die GUI von Kepler erweitert werden.

KAREntryHandler Erlaubt Modulen das Handling von KAR-Dateien, dem Dateiformat von Kepler-Workflows.

Authentication Framework Für die Authentifizierung und Autorisierung der Aktoren gegenüber der von ihnen eingebundenen Systeme, steht ein allgemeines und zentrales Authentication Framework zur Verfügung.

5.3 Funktionsumfang

In diesem Abschnitt wird auf die Evaluation der Kriterien im Zusammenhang mit dem Funktionsumfang des SDMCenter bzw. Kepler eingegangen.

5.3.1 Datenquellen

Bei der Installation von Kepler werden bereits eine Vielzahl von Aktoren mitgeliefert, mit denen auf gängige Datenquellen wie Datenbanken, Dateisysteme, FTP-Server, SSH-Server und Web Services zugegriffen werden kann. Es werden auch spezialisierte Systeme unterstützt die im wissenschaftlichen Bereich Anwendung finden, wie z.B. DiGIR (Distributed Generic Information Retrieval) [DIG], RBNB (Ring Buffer Network Bus) DataTurbine [RBN], Antelope ORB (Object Ring Buffer) [ANTa] und SDSC (Storage Resource Broker) Systeme.

5.3.2 Datenformate

Die unterstützten Datenformate variieren je nach Aktor und müssen vom Benutzer, je nach Anwendungsfall und verwendeter Aktoren zur Weiterverarbeitung, selbst ausgewählt werden. Somit können Daten von einer relationalen Datenbank beispielsweise als Arraystruktur, als Record oder als XML-Format abgerufen werden. Manche Aktoren bieten auch die Möglichkeit eigene Ports zu definieren. So können beispielsweise Aktoren realisiert werden, die mehrere verschiedene Datenformate zur gleichen Zeit unterstützen.

5.3.3 Datenzugriffstechnologien

Die unterstützten Datenzugriffstechnologien werden durch die Aktoren gekapselt und sind für den Benutzer nicht transparent. Durch die mitgelieferten Aktoren werden bereits verschiedene gängige Datenzugriffstechnologien unterstützt. Dazu gehören ODBC für Datenbanken, Web Services zur Integration verschiedenster Datenzugriffstechnologien, Zugriff auf Dateisysteme (lokal, per FTP und SSH) und der Zugriff auf spezialisierte Systeme über standardisierte Protokolle wie z.B. HTTP.

5.3.4 Datenverarbeitungsmöglichkeiten / Datenmanagement-Patterns

Die Aktoren von Kepler unterstützen in der Regel direkt die jeweilige Anfragesprache der unterstützten Datenquelle wie z.B. die Anfragesprache SQL bei relationalen Datenbanken. Daher hängt die Unterstützung der Datenmanagement-Patterns hauptsächlich von der Mächtigkeit der jeweiligen Anfragesprache bzw. der Unterstützung der jeweiligen Datenquelle ab. In der Regel werden für die verschiedenen Datenquellen grundlegende ETL-Patterns unterstützt, mit denen sich übergeordnete Patterns realisieren lassen. Zu diesem Zweck lassen sich in Kepler die Aktoren zu einem CompositeActor aggregieren, der wie ein einfacher Aktor für die Modellierung verwendet werden kann. Es können solche Patterns aber auch über Sub-Workflows realisiert werden. Für manche Datenquellen gibt es auch Aktoren mit Basic Data Management Patterns, wie z.B. den „Data Grid File Transfer"-Aktor, für den Transfer von Dateien zwischen lokalen und entfernten Dateisystemen.

5.3.5 Flexibilität zur Deploymentzeit

Modellierte Workflows müssen in Kepler nicht explizit deployt werden. Es gibt in der Toolbar eine Workflow-Steuerung, mit der sich der Workflow starten, pausieren und stoppen lässt. Der Ablauf des Workflows hängt dabei von den verwendeten Direktoren ab, die sich vor dem Starten bezüglich des Ablaufs einstellen lassen, wie z.B. das Festlegen der Anzahl von Iterationen.

5.3.6 Flexibilität zur Laufzeit

Eine Flexibilität zur Laufzeit ist in Kepler nicht gegeben, kann aber durch Erweiterungen realisiert werden. So können zum Beispiel Anforderungen an eine Datenquelle bei der Modellierung als Parameter eines Aktors definiert werden, die dann vom Aktor zur Laufzeit ausgewertet werden, um eine passende Datenquelle ausfindig zu machen.

5.3.7 Möglichkeit Anforderungen an Datenqualität zu formulieren

Kepler bietet keine Möglichkeit, Anforderungen an die Datenqualität zu formulieren. Dies lässt sich aber ebenfalls durch Erweiterungen realisieren.

5.3.8 Transparenz der Datenbereitstellung und des Datenmanagements

Die Transparenz der Datenbereitstellung und des Datenmanagements kann in Kepler über ein extra Provenance-Modul nachgerüstet werden, wird aber standardmäßig nicht unterstützt. Ebenfalls gibt es ein Reporting-Modul für das Monitoring, das nachgerüstet werden kann. Beide Module sind in einer Reporting-Suite für Kepler enthalten, die auch noch andere

Module enthält, mit denen Workflows für das Reporting semantisch getaggt (annotiert) werden können und Reportdokumente erstellt werden können.

5.4 Leistungsfähigkeit

Dieser Abschnitt beschreibt die Evaluation der Kriterien, die sich mit der Leistungsfähigkeit des SDMCenter bzw. Kepler befassen.

5.4.1 Performanz des Datenzugriffs

Verbindungen zu Datenquellen lassen sich bei der Modellierung durch Querverbindungen der Beziehungen zwischen den entsprechenden Aktoren wiederverwenden. Inwiefern dabei offene Verbindungen tatsächlich wiederverwendet werden und ob diese z.B. in einem Connection-Pool gehalten werden, lässt sich dabei nicht feststellen. Aktoren für den Verbindungsaufbau können aber grundsätzlich mit solchen Performance-Maßnahmen ausgerüstet werden.

5.4.2 Performanz der Datenverarbeitung

Die Performanz der Datenverarbeitung lässt sich bei Kepler nicht feststellen und ist stark vom Modellierer und der Implementierung der verwendeten Aktoren abhängig. Bei Aktoren, die DM-Befehle bzw. DM-Operationen direkt von den Datenquellen ausführen lassen, hängt die Performanz aber prinzipiell von der jeweiligen Datenquelle ab. Über entsprechende Aktoren ist hier auch ein direkter Transfer zwischen zwei Datenquellen realisierbar.

5.4.3 Performanz des Datentransfers

Eine Referenzierung von Daten wird von Kepler nicht unterstützt, ebenso gibt es keine Möglichkeit Daten explizit zu komprimieren. Die Performanz des Datentransfers hängt somit von der Implementierung und der Konfigurationsmöglichkeiten der verwendeten Aktoren ab.

5.4.4 Potentielle Optimierungsmöglichkeiten der Datenbereitstellung

Kepler bietet keine Optimierungsmöglichkeiten der Datenbereitstellung.

5.5 Anbindung und Integration

In diesem Abschnitt wird auf die Evaluation der Kriterien im Zusammenhang mit der Anbindung und Integration des SDMCenter bzw. Kepler eingegangen.

5.5.1 Integration von Werkzeugunterstützung

Kepler integriert alle Werkzeuge, die zur Modellierung, Ausführung und Auswertung von Workflows notwendig sind. Weitere Werkzeuge können über Erweiterungen realisiert werden.

5.5.2 Anbindungsmöglichkeiten an andere Systeme/Dienste

Durch die hohe Integration aller Werkzeuge in Kepler, gibt es keine Möglichkeit einzelne Teile von Kepler bzw. Kepler selbst an andere Systeme und Dienste anzubinden. Bei den Aktoren und Modulen handelt es sich jedoch um Java-Komponenten, die unter Umständen von anderen Java Anwendungen wiederverwendet werden können. Eine Zugriffsmöglichkeit auf Workflows von Außen, z.B. über eine Web Service Schnittstelle, ist nicht vorhanden.

5.5.3 Fähigkeit verschiedene wissenschaftliche Programme zu koppeln

Mit den Erweiterungsmöglichkeiten von Kepler ist es möglich Kepler mit jeder Art von wissenschaftlichen Programmen zu koppeln, die über eine Art von API verfügen. Es können Aktoren mit entsprechenden Konfigurationsmöglichkeiten für den Zugriff auf die Systeme realisiert werden, und ebenfalls Aktoren zur Datenverarbeitung und -transformation.

6 Fazit für SIMPL

In diesem Kapitel werden die untersuchten Systeme mit SIMPL anhand der in Kapitel 4 und 5 aufgezeigten Evaluationsergebnisse verglichen. Anhand der Evaluationsergebnisse sollen so die bereits vorhandenen Stärken von SIMPL sowie eventuelle Erweiterungsmöglichkeiten aufgezeigt werden.

6.1 Stärken von SIMPL

Um eine einheitliche Struktur beizubehalten, orientiert sich der nachfolgende Vergleich wieder an den Kriterien aus Kapitel 3. Dabei werden allerdings die Kategorien und Kriterien ausgeblendet, in denen es keine nennenswerte Unterschiede zwischen SIMPL und den beiden untersuchten Systemen gibt.

6.1.1 Datenquellen (siehe Kapitel 5.3.1)

SIMPL bietet im Vergleich zu Kepler eine generische Lösung für die Aktivitäten. Diese generische Lösung ermöglicht einen abstrahierten Zugriff auf die verschiedenen Datenquellen. Eine Erweiterung für eine neue Art von Datenquelle bedarf daher keiner weiteren Aktivitäten bzw. Aktoren, wie das bei Kepler der Fall ist. Dadurch bleiben die erstellten Workflows weitestgehend datenquellenunabhängig und ein Workflow kann beispielsweise durch einfache Konfiguration der Aktivitäten auf eine andere Art von Datenquelle umgestellt werden, ohne den Workflow selbst zu verändern, wie das bei neuen Aktivitäten bzw. Aktoren in Kepler der Fall wäre.

6.1.2 Datenverarbeitungsmöglichkeiten / Datenmanagement-Patterns (siehe Kapitel 4.3.4 und Kapitel 5.3.4)

OGSA-DAI ermöglicht momentan nur die Verarbeitung von Daten, die in OGSA-DAI oder einer Anwendung, die OGSA-DAI integriert, geladen wurden. D.h. Daten können nicht direkt von einer physischen Ressource zu einer anderen physischen Ressource ohne den Umweg über OGSA-DAI übertragen werden. Bei SIMPL beziehen sich die Datenmanagement-Patterns hauptsächlich auf die Verarbeitung von externen Daten. D.h. sofern es möglich ist, werden die Daten immer direkt, also ohne in SIMPL geladen zu werden, zwischen Ressourcen übertragen. In SIMPL werden Datenmanagement-Patterns weiterhin als integraler Bestandteil

des Rahmenwerks betrachtet, dies erlaubt im Gegensatz zu OGSA-DAI und dem SDMCenter eine abstraktere und einfachere Modellierung von Datenmanagement-Patterns.

6.1.3 Flexibilität zur Deploymentzeit (siehe Kapitel 4.3.5 und Kapitel 5.3.5)

Dadurch dass SIMPL an eine Workflow-Engine angebunden ist, liefert die Workflow-Technologie hier einige implizite Vorteile für die Flexibilität zur Deploymentzeit von SIMPL. So lassen sich beispielsweise zur Deploymentzeit des Workflows noch modellierte Datenquellenabfragen ändern.

6.1.4 Flexibilität zur Laufzeit (siehe Kapitel 4.3.6 und Kapitel 5.3.6)

Analog zum vorherigen Punkt bringt auch hier die Nutzung der Workflow-Technologie einige Vorteile mit sich, wie z.B. Late Binding von Datenquellen (siehe Kapitel 2.3).

6.1.5 Möglichkeit Anforderungen an Datenqualität zu formulieren (siehe Kapitel 4.3.7) und Kapitel 5.3.7

Auch in diesem Punkt liefert die Workflow-Technologie einige Vorteile. So kann beispielsweise sehr einfach ein Datenqualitäts-Framework (siehe [RBD+11]) in die Workflow-Engine integriert und so dessen Funktionalität in SIMPL genutzt werden.

6.1.6 Performanz des Datentransfers (siehe Kapitel 5.4.3)

SIMPL bietet, im Vergleich zu den anderen Systemen, die Möglichkeit zur Referenzierung von Daten in Verbindung mit einem Reference Resolution System (RRS), das in [WGSL09] beschrieben wird. Ebenfalls möglich ist die Referenzierung von Datenquellen über *Data Source Reference Variables* und die Referenzierung von Datencontainern über *Data Container Reference Variables*. Ein Datencontainer ist eine identifizierbare Sammlung von Daten, wie beispielsweise eine Tabelle in einer Datenbank oder eine Datei in einem Dateisystem [RRS+11].

6.1.7 Potentielle Optimierungsmöglichkeiten der Datenbereitstellung (siehe Kapitel 4.4.4 und Kapitel 5.4.4)

In SIMPL gibt es entsprechende integrierte Optimierungsmöglichkeiten bzw. solche können durch SIMPL genutzt werden (siehe [VSS+07]).

6.1.8 Integration von Werkzeugunterstützung (siehe Kapitel 4.5.1 und Kapitel 5.5.1)

Die Integration von SIMPL in ein Standardtool wie Eclipse (siehe Kapitel 2.3) stellt einen entscheidenden Vorteil gegenüber OGSA-DAI und SDMCenter bzw. Kepler dar. Eclipse liefert einen riesigen Fundus an Tools bzw. kann um weitere Tools ergänzt werden, die dadurch auch für SIMPL genutzt werden können, z.B. kann so ein SQL-Editor aus Eclipse für die Modellierung der SQL-Abfragen für SIMPL genutzt werden.

6.2 Identifizierte Erweiterungsmöglichkeiten für SIMPL

In den folgenden Kapiteln werden die identifizierten Erweiterungsmöglichkeiten für SIMPL beschrieben, die sich für SIMPL in Bezug auf die evaluierten Systeme herausgestellt haben.

6.2.1 In Bezug auf OGSA-DAI

SIMPL besitzt derzeit kein eigenes Event-Modell, sondern emittiert Ereignisdaten über das durch die Workflow-Engine bereitgestellte BPEL-Event-Modell. Das BPEL-Event-Modell liefert ein weitgehend ausreichendes Event-Modell für die Workflow-interne Datenverarbeitung (Verarbeitung der Workflow-Variablen). Hier wäre die Bereitstellung eines SIMPL-Event-Modells für das externe Datenmanagement (z.B. Datenquellenzugriffe, Datenquellenverbindungen oder Datenquellenabfragen) von Vorteil, das auf die Datenintegration und das Datenmanagement zugeschnitten ist und detaillierte Informationen liefern kann. Dazu gehört ebenfalls die Implementierung eines entsprechenden Event-Frameworks, das SIMPL-Events verarbeitet, persistiert und für Anwendungen, die SIMPL integrieren, zugänglich macht. Dies würde die Transparenz von SIMPL erhöhen und Ereignisdaten unabhängig von der Integration von SIMPL machen. D.h. die Ereignisdaten stehen sowohl einer Workflow-Engine als auch einer klassischen Java-Anwendung über ein und dieselbe Schnittstelle zur Verfügung.

Zur (asynchronen) Ausführung von komplexen OGSA-DAI Daten-Workflows könnte OGSA-DAI an SIMPL angebunden werden. So könnten beispielsweise die Vorbereitungsschritte einer Simulation (Datenabfragen, Erstellung von Containern (Dateien, Tabellen, ...), Datentransfer, ...), die Transformation großer Datenmengen oder auch der Transfer großer Datenmengen asynchron zum Kontrollfluss der Ausführung der Workflow-Instanz (BPEL) realisiert werden. Ein entsprechender Anwendungsfall wäre beispielsweise die in [Dor11] beschriebenen Kopplung der Simulationsanwendungen PANDAS (Porous Media Adaptive Nonlinear finite element solver based on Differential Algebraic Systems, [PAN]) und Matlab. So könnte man z.B. die Daten, die PANDAS generiert, über OGSA-DAI Workflows an die Rechner übertragen, auf denen die einzelnen Matlab-Instanzen ausgeführt werden, und natürlich auch umgekehrt die Ergebnisdaten der Matlab-Instanzen zurück zu PANDAS übertragen. SIMPL könnte dabei zwischen dem Nutzer und OGSA-DAI vermitteln, so dass die meisten Informationen, die OGSA-DAI als Eingabe benötigt, durch SIMPL generiert oder geliefert werden könnten, ohne die vorhandene Schnittstelle zum Nutzer ändern zu

müssen. Analog dazu könnte man natürlich auch direkt durch den Nutzer modellierte OGSA-DAI Workflows über SIMPL zur Verarbeitung an OGSA-DAI weiterleiten. So könnten beispielsweise die Stärken der workflow-basierten Simulationsausführung sowie von SIMPL und OGSA-DAI kombiniert werden. In [Age11] wurde das System *Champagne*, ein System für die Propagation von Datenänderungen, als weitere Möglichkeit zur Verwendung für einen solchen Einsatzzweck vorgeschlagen. Hierzu könnte man beide Systeme (OGSA-DAI und Champagne) sowie die herkömmliche Variante über den expliziten Kontrollfluss der BPEL-Workflows vergleichen (siehe [Dor11]).

Ebenfalls eine interessante Erweiterungsmöglichkeit für SIMPL wäre die in Kapitel 4.4.2 beschriebene Integration von Apache Lucene, um Dateien bzw. deren Inhalt zu indizieren und Dateien so effizienter durchsuchen zu können.

Die in Kapitel 4.2.2 beschriebene Möglichkeit zur Registrierung neuer Datenquellen und zum automatischen Deployment deren Treiber zur Laufzeit des Servers wäre auch für SIMPL eine interessante Erweiterung. Im Moment müssen die Treiber noch von Hand deployed und Apache Tomcat jedesmal neu gestartet werden. Stattdessen könnte man die Registrierung der Datenquellen im Web Interface des Resource Management vereinfachen. Über das Web Interface könnte man so z.B. bei SQL-Datenbanken die Pfade zu den entsprechenden JAR-Dateien der JDBC-Treiber angeben, und dann werden diese automatisch in Apache Tomcat deployed.

6.2.2 In Bezug auf SDMCenter

Einen Vorteil gegenüber SIMPL bietet Kepler bei der Auswahl an unterstützten Datenformaten. So können an einem Aktor mehrere Output-Ports mit unterschiedlichen Datenformaten definiert werden und dadurch parallel verschiedene Datenformate weiterverarbeitet werden. Das ist z.B. in XML, als Record, oder als Arraystruktur möglich. In SIMPL ist das Datenformat einer Datenmanagement-Aktitäten ausschließlich ein XML-Datenformat, das von dem zuständigen Connector und Converter der in der Datenmanagement-Aktivität ausgewählten Datenquelle, abhängig ist. Dies ist immer der Fall, egal ob man die Daten in die Workflow-Engine lädt oder ob man sie im SIMPL Core bzw. im Service Bus verarbeitet. Da auch große Datenmengen in SIMPL in ein XML-Datenformat transformiert werden könnten, kann an dieser Stelle SIMPL verbessert werden, indem man Möglichkeiten schafft mit den im SIMPL Core bzw. im Service Bus nativen Datenformaten (Java-Datentypen) der Connectoren zu arbeiten und nur bei Bedarf, also bei der Verarbeitung der Daten innerhalb des Workflows, diese in ein XML-Datenformat zu transformieren.

Kepler und SIMPL bieten beide ein zentrales Konfigurationsmanagement. Kepler bietet zudem ein zentrales, allgemeines Authentifizierungs-Framework, das es Aktoren ermöglicht und erleichtert, sich gegenüber den zu integrierenden Systemen zu authentifizieren und zu autorisieren. In SIMPL wird dagegen die Funktionalität der Authentifizierung und Autorisierung in den Connectoren realisiert, was spätestens bei der Unterstützung verschiedener Authentifizierungs- und Autorisierungstechnologien zu einer hohen Anzahl von redundanten und schwer wartbaren Implementierungen führt. Hier müsste man untersuchen, in

wie weit sich SIMPL an das Authentifizierungs- und Autorisierungsframework von Kepler koppeln lässt, oder ein eigenes Authentifizierungs-Framework, z.B. über das Resource Management, realisieren lässt.

SIMPL fehlen außerdem Möglichkeiten der Datenanalyse, die durch entsprechende Datenmanagement-Patterns und BPEL-Erweiterungs-Aktivitäten, für die Analyse geschaffen werden könnten.

Die weiteren Technologien von SDMCenter der DMA- und SEA-Datenmanagementebene sind für eine Verwendung in SIMPL größtenteils nicht geeignet, sondern finden viel mehr Anwendung in Systemen, die an SIMPL als Datenquelle angebunden werden können. Eine Technologie der DMA-Datenmanagementebene, die in SIMPL eingesetzt werden könnte, ist FastBit. Mit FastBit können die Daten des Resource Managements noch effizienter verwaltet werden, um auch für eine Vielzahl an Ressourcen vorbereitet zu sein und weiterhin einen schnellen Zugriff auf diese gewährleisten zu können.

7 Zusammenfassung und Ausblick

Die Evaluationen von OGSA-DAI und SDMCenter und der Vergleich mit SIMPL haben gezeigt, dass SIMPL ein gleichwertiges bzw. in einigen Aspekten sogar ein besseres Datenmanagementsystem ist. Durch die Nutzung der Workflow-Technologie bringt SIMPL einige implizite Vorteile für Erweiterbarkeit und die Flexibilität zur Deploymentzeit und Laufzeit mit sich. Ein weiterer Vorteil von SIMPL ist Betrachtung der Datenmanagement-Patterns als integraler Bestandteil des Rahmenwerks. Dies erlaubt im Gegensatz zu OGSA-DAI und dem SDMCenter eine abstraktere und einfachere Modellierung von Datenmanagement-Patterns. Ebenfalls ein entscheidender Vorteil ist die Werkzeugunterstützung von SIMPL durch Eclipse. Die „Defizite“ von SIMPL im Bezug auf die beiden untersuchten Systeme liegen lediglich an dessen Funktionsumfang. So ist beispielsweise die Auswahl an unterstützten Datenformaten geringer oder die Registrierung neuer Datenquellen in SIMPL nicht zur Laufzeit möglich. Die fehlende Funktionalität lässt sich aber relativ einfach durch die modulare Architektur von SIMPL einbinden, indem einfach neue Komponenten an SIMPL angebunden werden, die eine entsprechende Funktionalität liefern.

Ausblick

Die beiden untersuchten Systeme sind sehr umfangreich und teilweise auch komplex, daher gibt es eine Reihe weiterer Aspekte, die detaillierter untersucht und mit SIMPL verglichen werden könnten. Beispielsweise die Bereitstellung und Registrierung von Ressourcen in OGSA-DAI oder die Definition neuer Aktoren in SDMCenter. Dabei wäre vor allem auch die Mächtigkeit des gewählten Ansatzes (Ressourcen in OGSA-DAI bzw. Aktoren in SDMCenter) und dessen Einschränkungen interessant. Ein weiterer Aspekt, der detaillierter untersucht werden sollte, ist die Anbindung von Systemen für das Datenmanagement und die Datenbereitstellung an SIMPL. Interessant wäre vor allem, in welchen Szenarien eine Anbindung sinnvoll ist und wann eine Anbindung sich nicht lohnt. Ein entsprechendes Beispielszenario für die Anbindung von OGSA-DAI oder auch Champagne wurde in Kapitel 6 aufgezeigt. Dabei geht es um die Kopplung der Simulationsanwendungen PANDAS und Matlab. Mithilfe von SIMPL und OGSA-DAI oder Champagne könnte man z.B. die Daten, die PANDAS generiert, über OGSA-DAI Workflows an die Rechner übertragen, auf denen die einzelnen Matlab-Instanzen ausgeführt werden, und natürlich auch umgekehrt die Ergebnisdaten der Matlab-Instanzen zurück zu PANDAS übertragen.

Da es noch eine Reihe weiterer Systeme für das Datenmanagement und die Datenbereitstellung gibt, könnten analog zu dieser Fachstudie weitere Systeme wie beispielsweise Microsoft Trident [TRI] untersucht werden.

Literaturverzeichnis

[ADI] ADIOS. URL http://www.olcf.ornl.gov/center-projects/adios/. (Zitiert auf Seite 20)

[ADO] Adobe Flash. URL http://www.adobe.com/de/products/flash.html. (Zitiert auf Seite 21)

[Age11] C. Ageu. Kombination von SIMPL mit einem Ansatz zur Propagation von Datenänderungen. Studienarbeit: Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Anwendersoftware, 2011. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=STUD-2328&engl=0. (Zitiert auf Seite 60)

[ANTa] Antelope ORB. URL http://eqinfo.ucsd.edu/faq/antelope.php. (Zitiert auf Seite 52)

[ANTb] Apache Ant. URL http://ant.apache.org/. (Zitiert auf Seite 38)

[AXI] Apache Axis. URL http://axis.apache.org/. (Zitiert auf Seite 38)

[CEMP11] I. B. M. Corporation, T. U. of Edinburgh, U. P. de Madrid, K. Pradeeban. *OGSA-DAI 4.1 Documentation*, 2011. URL http://ogsa-dai.sourceforge.net/documentation/ogsadai4.1/ogsadai4.1-axis/. (Zitiert auf den Seiten 7, 11, 12, 14, 16, 17, 39, 40, 41, 43, 44 und 47)

[DBC] Apache Commons DBCP. URL http://commons.apache.org/dbcp/. (Zitiert auf Seite 46)

[DIG] DiGIR. URL http://digir.net/. (Zitiert auf Seite 52)

[Dor11] R. Dormien. *Service-Bus-Erweiterung um Pandas-basierte Simulationen in Workflows zu nutzen*. Diplomarbeit, Universität Stuttgart, Fakultät Informatik, Elektrotechnik und Informationstechnik, Deutschland, 2011. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=DIP-3127&engl=0. (Zitiert auf den Seiten 59 und 60)

[EML] EML. URL http://knb.ecoinformatics.org/software/eml/. (Zitiert auf Seite 22)

[ESS] eSimMon. URL http://www.olcf.ornl.gov/center-projects/esimmon/. (Zitiert auf Seite 21)

[FAS] FastBit. URL https://sdm.lbl.gov/fastbit/. (Zitiert auf Seite 21)

[GLO] Globus Toolkit. URL http://www.globus.org/toolkit/. (Zitiert auf Seite 38)

[GRA] Graphviz - Graph Visualization Software. URL http://www.graphviz.org/. (Zitiert auf den Seiten 39 und 45)

[KEP] Kepler. URL https://kepler-project.org/. (Zitiert auf Seite 21)

[LUC] Apache Lucene. URL http://lucene.apache.org/. (Zitiert auf Seite 47)

[MPIa] MPI. URL http://www-unix.mcs.anl.gov/mpi/mpi-standard/mpi-report-2.0-sf/mpi2-report.htm. (Zitiert auf Seite 19)

[MPIb] MPICH2. URL http://www.mcs.anl.gov/research/projects/mpich2/. (Zitiert auf Seite 19)

[NET] NetCDF. URL http://www.unidata.ucar.edu/software/netcdf/. (Zitiert auf Seite 20)

[PAN] PANDAS. URL http://www.mechbau.uni-stuttgart.de/pandas/index.php. (Zitiert auf Seite 59)

[PNE] Parallel-NetCDF. URL http://trac.mcs.anl.gov/projects/parallel-netcdf/. (Zitiert auf Seite 20)

[PRO] ProRata. URL http://code.google.com/p/prorata/. (Zitiert auf Seite 21)

[PVF] PVFS. URL http://www.pvfs.org/. (Zitiert auf Seite 20)

[RBD+11] M. Reiter, U. Breitenbücher, S. Dustdar, D. Karastoyanova, F. Leymann, H.-L. Truong. A Novel Framework for Monitoring and Analyzing Quality of Data in Simulation Workflows. In *2011 Seventh IEEE International Conference on eScience*. IEEE, 2011. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2011-77&engl=0. (Zitiert auf den Seiten 33 und 58)

[RBN] RBNB DataTurbine. URL http://www.dataturbine.org/. (Zitiert auf Seite 52)

[RM11] P. Reimann, B. Mitschang. Data Provisioning for Scientific Workflows. Poster-Präsentation im 4. SimTech Status Seminar, Bad Boll, Deutschland, 21.-23. November 2011. (Zitiert auf den Seiten 7 und 32)

[ROM] ROMIO. URL http://www.mcs.anl.gov/romio/. (Zitiert auf Seite 19)

[RPR] R. URL http://www.r-project.org/. (Zitiert auf Seite 21)

[RRS+11] P. Reimann, M. Reiter, H. Schwarz, D. Karastoyanova, F. Leymann. SIMPL - A Framework for Accessing External Data in Simulation Workflows. In G. für Informatik (GI), editor, *Datenbanksysteme für Business, Technologie und Web (BTW 2011), 14. Fachtagung des GI-Fachbereichs „Datenbanken und Informationssysteme“ (DBIS), Proceedings, 02.-04. März 2011, Kaiserslautern, Deutschland*, Series of the Gesellschaft für Informatik (GI), pp. 534–553. Lecture Notes in Informatics (LNI), 2011. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/

NCSTRL/NCSTRL_view.pl?id=INPROC-2011-07&engl=. (Zitiert auf den Seiten 34 und 58)

[SAP] Sapphire. URL https://computation.llnl.gov/casc/sapphire/. (Zitiert auf Seite 21)

[SCI] SciDAC. URL http://www.scidac.gov/. (Zitiert auf Seite 17)

[SDM] SDMCenter. URL https://sdm.lbl.gov/sdmcenter/. (Zitiert auf den Seiten 7, 17 und 18)

[SRM] SRM-Lite. URL https://sdm.lbl.gov/srmlite/. (Zitiert auf Seite 20)

[TOM] Apache Tomcat. URL http://tomcat.apache.org/. (Zitiert auf Seite 38)

[TRI] Microsoft Trident. URL http://research.microsoft.com/en-us/collaboration/tools/trident.aspx. (Zitiert auf Seite 63)

[VSS+07] M. Vrhovnik, H. Schwarz, O. Suhre, B. Mitschang, V. Markl, A. Maier, T. Kraft. An Approach to Optimize Data Processing in Business Processes. In *Proc. of the 33rd International Conference on Very Large Data Bases (VLDB 2007), Vienna, Austria, September 23-28, 2007*, pp. 615–626. 2007. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2007-28&engl=. (Zitiert auf den Seiten 34 und 58)

[WGSL09] M. Wieland, K. Görlach, D. Schumm, F. Leymann. Towards Reference Passing in Web Service and Workflow-based Applications. In *Proceedings of the 13th IEEE Enterprise Distributed Object Conference (EDOC 2009)*, pp. 109–118. IEEE, Auckland, New Zealand, 2009. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=INPROC-2009-52&engl=0. (Zitiert auf Seite 58)

Alle URLs wurden zuletzt am 14.02.2012 geprüft.

Erklärung

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

(Michael Hahn Michael Schneidt)