Forschungsdaten-Föderations-Architektur (Prototyp) (M 4.3.4.1)

Version 10.11.2017
Cluster 4
Verantwortlicher Partner MInf-BA

DARIAH-DE

Überführung der digitalen Forschungsinfrastrukturen für die e-Humanities in die Operational Phase (Betriebsphase)

Dieses Forschungs- und Entwicklungsprojekt wird / wurde mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF), Förderkennzeichen 01UG1610A bis J, gefördert und vom Projektträger im Deutschen Zentrum für Luft- und Raumfahrt (PT-DLR) betreut.

GEFÖRDERT VOM

Bundesministerium für Bildung und Forschung

Projekt: DARIAH-DE: Überführung der digitalen Forschungsinfrastrukturen für die e-Humanities in die Operational Phase (Betriebsphase)

BMBF Förderkennzeichen: 01UG1610A bis J
Laufzeit: März 2016 bis Februar 2019

Dokumentstatus: Final

Verfügbarkeit: öffentlich

Autoren:

Tobias Gradl, MInf-BA

Revisionsverlauf:

Datum	Autor	Kommentare
10.11.2017	Tobias Gradl	Entwurf finalisiert

Inhaltsverzeichnis:

1. Einleitung

Die Digitalisierung ist ein Themenfeld, das in unterschiedlichen Bereichen von Wissenschaft und Industrie eine lange Historie aufweisen kann. Begünstigt durch die Diskussion in Politik und Medien gewann das Thema in jüngerer Vergangenheit zusätzlich an Aktualität und wird nicht zuletzt auch im Bereich der Kultur- und Geisteswissenschaften verstärkt gefördert. Tatsächlich sind die Digitalisierung und die damit verbundene digitale Edition und Annotation grundlegende Faktoren für den Erhalt und die Bereitstellung von Forschungsobjekten und -daten mit Hilfe digitaler Methoden. Eine populäre, intuitive Annahme besteht darin, dass durch eine erfolgte Digitalisierung und eine digitale Bearbeitung die notwendigen Schritte auch für die weiterführende Nutzung dieser Daten abgeschlossen sind. Tatsächlich stehen die Akteure der digitalen Welt in einem ständigen Spannungsfeld zwischen Informationsüberfluss und fehlender Zugänglichkeit von Information. Digitale Infrastrukturen helfen bei unterschiedlichen Aspekten: übergreifende Perspektiven auf Daten und Metadaten unterstützen beispielsweise deren Analyse und Visualisierung, Suchmaschinen ermöglichen oft erst die Auffindbarkeit relevanter digitaler Artefakte.

Um über lokale Nutzen hinaus eine Nachnutzbarkeit kultur- und geisteswissenschaftlich relevanter Forschungsdaten zu unterstützen, bietet DARIAH-DE Methoden und Werkzeuge zur Publikation, Referenzierung, Modellierung und Assoziation von Forschungsdaten. Unter der Forschungsdaten-Föderationsarchitektur (DFA[1]) werden derzeit die Collection Registry (CR), das Repository, der Publikator, das Data Modeling Environment (DME), die Generische Suche und der Epic-PID Service zusammengefasst.[2] Ihr modularer Charakter und die angebotenen Schnittstellen der einzelnen Komponenten ermöglichen die Erweiterung der DFA um weitere Methoden und Werkzeuge.

Flankiert durch weitere aktuelle Berichte[3] aus Cluster 4, wie das Schnittstellenkonzept (M 4.3.1), das Konzept für ein Schulungs- und Dokumentationsportal (M 4.3.5.1) und den Prototyp des DARIAH-DE Repositoriums (M 4.3.3.1) fokussiert der vorliegende Meilensteinbericht auf die CR und DME und begleitet deren Installation.

---

[1] Häufig kurz: *Datenföderationsarchitektur*
[2] Siehe auch https://de.dariah.eu/data-federation-architecture
[3] Siehe https://wiki.de.dariah.eu/display/publicde/Reports+and+Milestones

2. Komponentenübersicht

Abbildung 1 skizziert wesentliche Bausteine und deren Beziehungen innerhalb der DFA. Die dargestellte Übersicht ist eine Adaption des ursprünglichen DFA Architekturmodells (Gradl et al., 2015) und führt die Komponente des DME als Weiterentwicklung der DARIAH-DE Schema Registry (SR) ein.

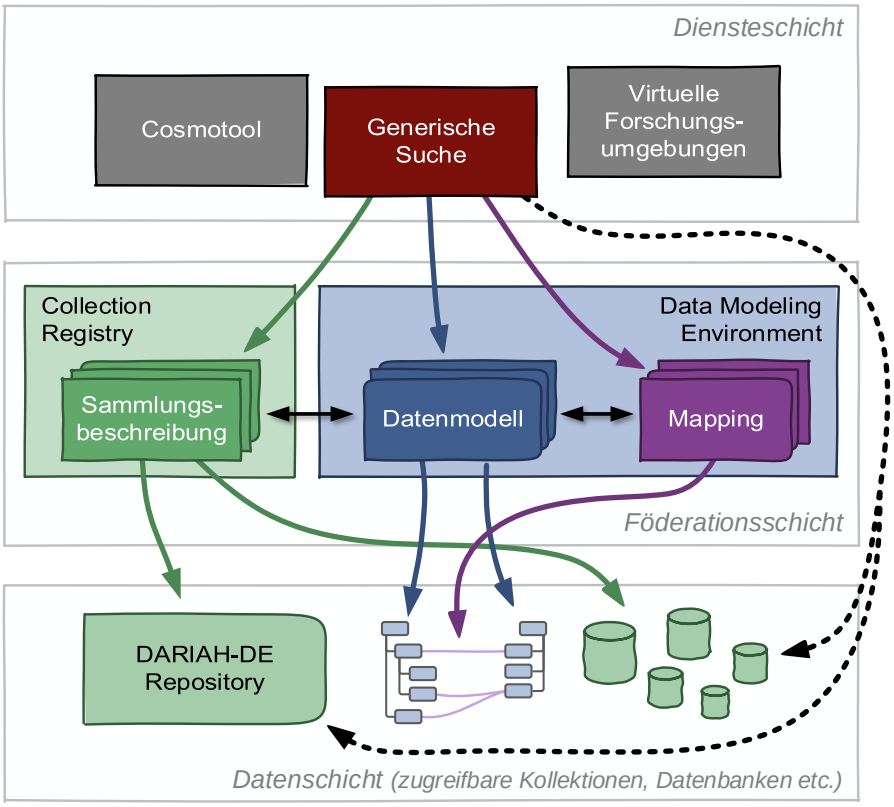


Abbildung 1: Komponenten und Beziehungen in der DARIAH-DE

Die Komponenten der DFA können anhand ihrer wesentlichen Funktionalität der *Datenschicht*, *Föderationsschicht* oder *Diensteschicht* zugeordnet werden.

In der ***Datenschicht*** werden im Kontext der DFA solche Komponenten zusammengefasst, die einen maschinellen Zugriff auf Daten über Schnittstellen erlauben und dadurch für Dienste übergeordneter Schichten zugreifbar sind. Angebotene Schnittstellen sind zumeist web- und XML-basiert. Das im Bereich digitaler Sammlungen häufig verwendete OAI-PMH[4] oder über das Internet zugängliche XML-Dateien sind typische Zugriffsschnittstellen. DARIAH-DE bietet daneben eine eigene Repository-Lösung für die Publikation von Forschungsdaten an.

[4] Open Archives Initiative Protocol for Metadata Harvesting: https://www.openarchives.org/pmh/

Die ***Föderationsschicht*** besteht aus den Komponenten der CR und des DME. Beide Dienste verwalten jeweils unterschiedliche Objektarten, ermöglichen jedoch in der Kombination der jeweils beinhalteten Informationen die Erstellung integrativer Sichten über heterogene Daten. Sammlungsbeschreibungen der CR umfassen neben unterschiedlichen weiteren Aspekten[5] insbesondere auch Verweise auf die für eine Kollektion jeweils gültigen Datenmodelle im DME. Durch diese Verbindung werden Daten der beschriebenen Kollektion im Rahmen der DFA interpretierbar. Das DME verzeichnet neben einzelnen Datenmodellen auch – soweit modelliert – deren Zusammenhänge (Abbildungen, Mappings) und bildet so die Basis für eine Zusammenführung (Föderation) modellkonformer Daten.

Komponenten der ***Diensteschicht*** nutzen die Informationen untergeordneter Schichten und zielen im Wesentlichen darauf ab, einen Nutzen[6] für fachwissenschaftliche Anwenderinnen und Anwender zu generieren. In Bezug auf die Idee der DFA bedeutet dies typischerweise die Bereitstellung heterogener Daten in Form harmonisierter Sichten. Die in der Abbildung hervorgehobene generische Suche nutzt die in Sammlungsbeschreibungen verzeichneten Schnittstellen für den Zugriff auf Kollektionen. Mit Hilfe der in der DME hinterlegten Datenmodelle können die bezogenen Daten verarbeitet, interpretiert und schließlich indexiert werden. Zum Anfragezeitpunkt erlauben Abbildungen zwischen den Datenmodellen eine Definition von Suchparametern anhand einzelner Datenmodelle, welche in äquivalente Konzepte anderer Datenmodelle übersetzt werden.

3. Technische Architektur

Als weitere Perspektive auf die DFA stellt Abbildung 2 die bislang implementierten und verwendeten Komponenten und Bibliotheken aus dem technischen Blickwinkel der Softwareentwicklung zusammen. Für eine verbesserte Abgrenzbarkeit sind die Komponenten unterschiedlich eingefärbt: Eigenständig lauffähige, von DARIAH-DE entwickelte *Dienste* sind *blau*, die in DARIAH-DE entwickelten *Bibliotheken grün*, sowie die verwendeten Fremdbibliotheken *grau* eingefärbt. Um das Diagramm nicht zu überladen werden nur die für das Verständnis der Funktionsweise relevanten Bibliotheken

[5] siehe DARIAH Collection Description Data Model (DCDDM), https://github.com/DARIAH-DE/DCDDM
[6] Die Modellierung von Daten mit Hilfe des DME erzielt durch die intensive Auseinandersetzung mit den Kollektionen auch einen direkten Nutzen. Sie dient dennoch typischerweise nicht diesem Selbstzweck, sondern der Vorbereitung der Föderation durch weiterverarbeitende Dienste.

dargestellt. Alle Abhängigkeiten können in den jeweiligen Project Object Model (POM)[7] Dateien der Komponenten eingesehen werden. Für die CR[8] und das DME[9] sind diese direkt in GitHub verfügbar, alle weiteren Komponenten sind (auch im Quellcode) über ein dediziertes Artifactory Softwarerepository[10] öffentlich zugänglich.

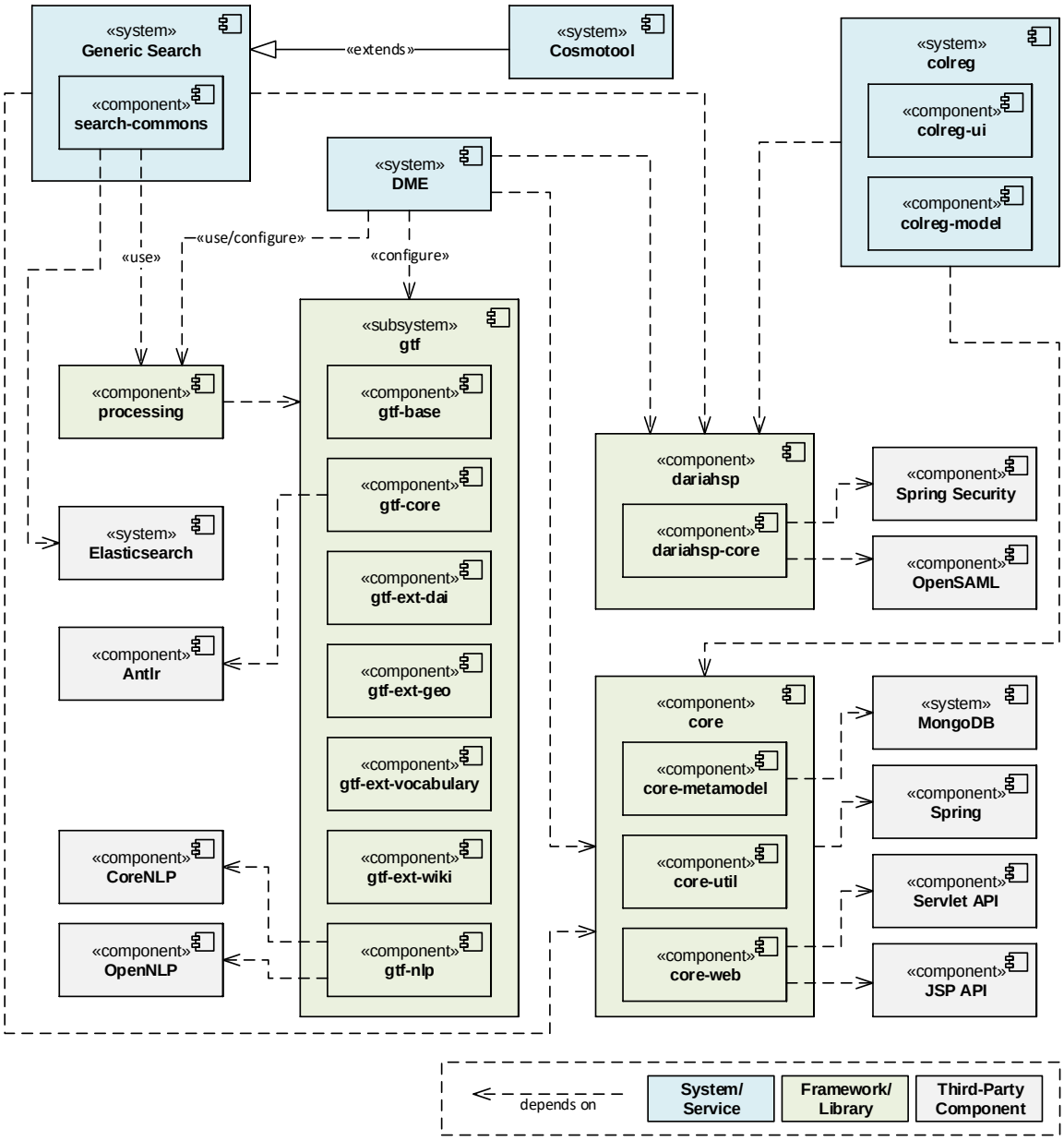


Abbildung 2: Komponentendiagramm Gesamtübersicht

[7] Konzept von Apache Maven: https://maven.apache.org/
[8] https://github.com/DARIAH-DE/colreg/blob/master/pom.xml
[9] https://github.com/tgradl/dme/blob/master/pom.xml
[10] Einstieg über https://minfba.de.dariah.eu/artifactory, für z. B. core-metamodel: https://minfba.de.dariah.eu/artifactory/webapp/#/artifacts/browse/tree/General/dariah-minfba-snapshots/de/unibamberg/minf/core/core-metamodel

- Sämtliche Dienste basieren auf Grundfunktionalität der `core` Bibliotheken: Das Metamodell für Datenmodelle, Mappings etc. findet sich in `core-metamodel`, `core-util` implementiert Hilfsklassen und `core-web` bietet einen gemeinsamen technischen Rahmen für die Web-Anwendungen auf Basis von Spring MVC[11], der Servlet-[12] und JSP-API[13].
- Das Grammatical Transformation Framework (`gtf`) implementiert die Methoden der Datendefinition und Datentransformation und damit die grundlegende Idee der forschungsorientierten Föderation (vgl. 4.2.1). `gtf-base` und `gtf-core` kapseln basale Modellklassen und Funktionalität. Weiterführende Transformationslogik wird im Rahmen von Erweiterungsbibliotheken implementiert. Die derzeitigen Erweiterungen und jeweils verwendete externe Bibliotheken sind in der Abbildung dargestellt.
- Das GTF ist generisch implementiert und auch für andere Szenarien einsetzbar. Die Bibliothek `processing` bildet einen Wrapper für die Funktionalität des GTF für die DARIAH-DE DFA.
- Das DME ist als Konfigurationsoberfläche für `gtf` und `processing` zu verstehen, nutzt beide Komponenten darüber hinaus auch selbst für deren Anwendung auf Beispieldaten (vgl. 4.3).
- Grundlegende Suchfunktionalität der Generischen Suche ist in `search-commons` zusammengefasst. Letzteres kapselt u. a. den Zugriff auf `processing` und `elasticsearch`. Das Cosmotool basiert in Bezug auf die DFA auf der Generischen Suche und teilt deren diesbezügliche Eigenschaften.
- Sämtliche Dienste nutzen `dariahsp`[14] als SAML Implementierung zur Anbindung an die DARIAH-DE AAI bzw. DFN AAI.

[11] https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html
[12] https://github.com/javaee/javaee-jsp-api
[13] https://github.com/javaee/servlet-spec
[14] https://github.com/tgradl/dariahsp

4. Prototypen

Gemeinsam mit diesem Dokument werden aktuelle Prototypen von CR und DME für den produktiven Betrieb veröffentlicht. Im Wesentlichen bedeutet dies, dass jeweils stabile Versionen durch eine neu eingerichtete virtuelle Maschine (https://dfa.de.dariah.eu) bereitgestellt werden.

- Im Fall der CR ist aus Sicht der Anwenderinnen und Anwender kein Unterschied feststellbar – diese ist weiterhin unter der bekannten URL https://colreg.de.dariah.eu verfügbar.
- Das DME löst die bisherige SR ab und steht unter https://dme.de.dariah.eu zur Verfügung. Aufrufe der URL https://schereg.de.dariah.eu werden entsprechend umgeleitet.

Neben der stabilen Installation werden jeweilige Vorabversionen der DFA Komponenten auf einer weiteren, eigens eingerichteten virtuellen Maschine installiert. Diese ist unter der URL https://dfatest.de.dariah.eu (Abbildung 3) erreichbar.

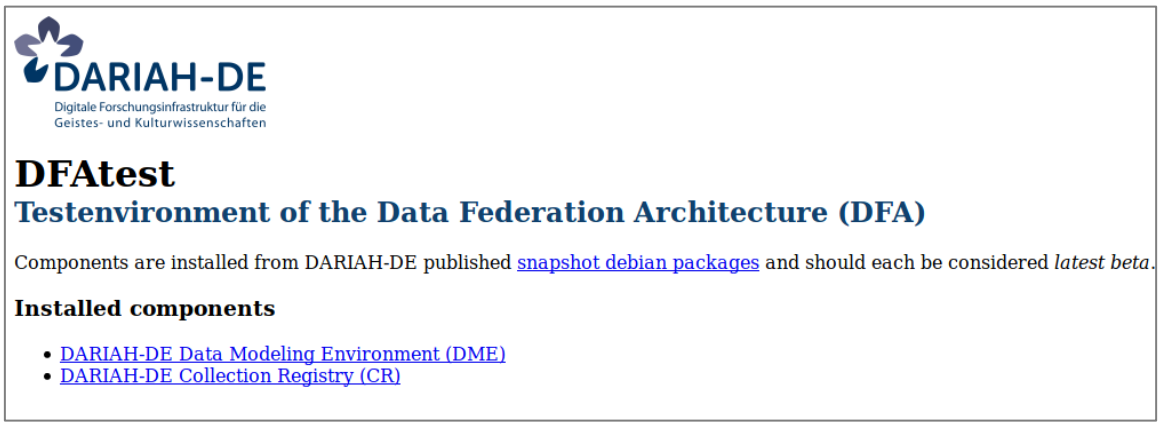


Abbildung 3: Landing Page der DFAtest Installation

Die Produktiv- und Testinstallationen basieren auf dem DARIAH-DE Debian Repository[15], welches eine besonders einfache Installation von Paketen unter Debian/Ubuntu ermöglicht. Die produktiven Dienste (DFA) werden durch *Release-Versionen* aktualisiert. Die Vorabversionen (DFAtest) basieren auf *Snapshot-Versionen*, dienen dem Test neuer Funktionalität und können auch Instabilitäten aufweisen. Sämtliche Daten der Testinstallationen sind losgelöst vom Produktivsystem und können jederzeit auch gelöscht werden. Weiterführende Hinweise finden sich auch in den Installationsanweisungen der Komponenten in den jeweiligen GitHub Repositories.

[15] https://ci.de.dariah.eu/packages/

4.1. Collection Registry (CR)

Zahlreiche Rückmeldungen und Erkenntnisse zu vorausgegangenen Versionen der CR führten zu einer vollständigen Neuentwicklung ab Januar 2016. Die CR (Abbildung 4) basiert seitdem – wie auch aus dem Komponentendiagramm in Abbildung 2 ersichtlich – auf der Softwarearchitektur weiterer DFA Komponenten.

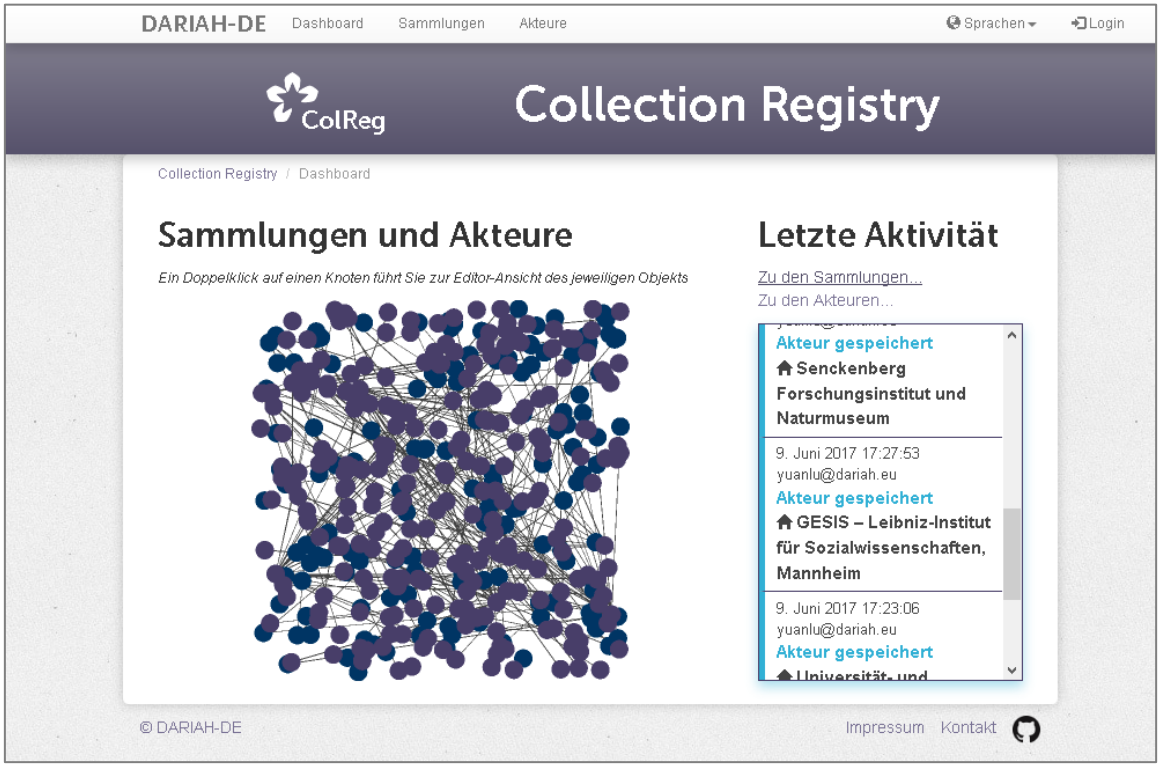


Abbildung 4: Aktuelle Startseite der Collection Registry (Stand: 2. November 2017)

4.1.1. Dokumentation

In unterschiedlichen Publikationen zur DFA wird auch die CR thematisiert (vgl. Gradl et al., 2015 und Gradl/Henrich, 2016b). Darüber hinaus wurden einige weiterführende Dokumente erarbeitet, die den Einstieg in die CR – insbesondere aus der Sicht fachwissenschaftlicher Anwenderinnen und Anwender - erleichtern:

- Auf der Portalseite[16] finden sich einige wichtige Fragen und Antworten zur CR.
- Der ausführliche User-Guide[17] stellt die wichtigsten Funktionalitäten der CR zusammen und erläutert deren Anwendung anhand von Bildschirmausschnitten.

[16] https://de.dariah.eu/collection-registry
[17] https://wiki.de.dariah.eu/display/publicde/Die+Collection+Registry

- Die Best-Practice-Empfehlungen[18] erleichtern den Einstieg in das Datenmodell der CR.
- Das Datenmodell[19] der CR steht selbst ebenfalls auf GitHub zur Verfügung.
- Das GitHub Repository[20] der CR beinhaltet eine ausführliche Installationsanleitung für diejenigen Nutzerinnen und Nutzer, die eine eigene Instanz installieren möchten.

4.1.2. Derzeitiger Stand

Die Codebasis der CR kann seit Anfang 2017 als stabil bezeichnet werden. Änderungen im Code fanden im Wesentlichen statt, um die Veränderungen in Schnittstellen anderer DFA Komponenten widerzuspiegeln.

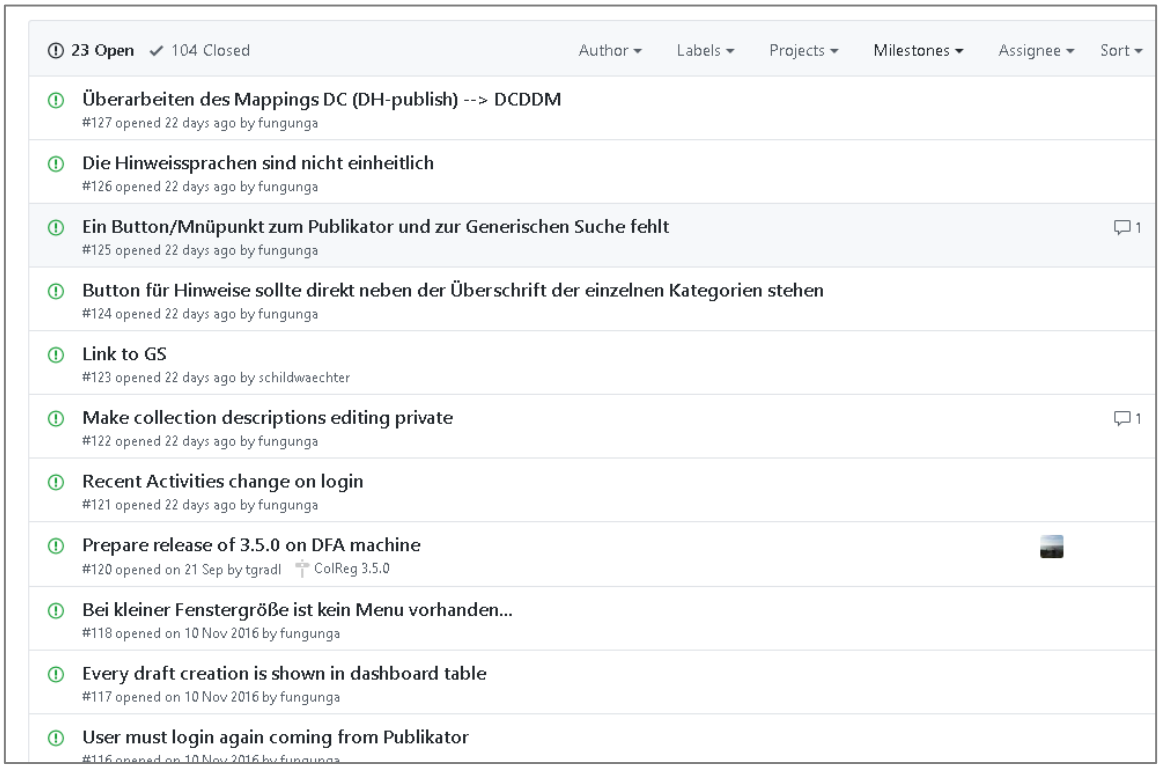


Abbildung 5: Auszug der aktuellen GitHub Issues (Stand: 2. November 2017)

Die derzeit implementierte Funktionalität wurde innerhalb der vergangenen Monate ausgiebig getestet und hat sich als recht robust erwiesen. GitHub Issues[21] zur CR sowie auch Anforderungen aus assoziierten Studien und Projekten, die bereits auf der

[18] https://wiki.de.dariah.eu/display/publicde/Best-Practice-Empfehlungen
[19] https://github.com/DARIAH-DE/DCDDM
[20] https://github.com/DARIAH-DE/colreg/blob/master/README.md
[21] https://github.com/DARIAH-DE/colreg/issues

CR basieren[22], adressieren einige Fehler und insbesondere auch so genannte *Feature Requests*, etwa zur verbesserten Suche in Kollektionsbeschreibungen. Bereits hinterlegte und aufkommende Rückmeldungen werden mindestens bis zum Ende der Projektlaufzeit von DARIAH-DE eingearbeitet. Die Bedeutung der DFA für DARIAH-DE sowie derzeit aufkommende Projektvorhaben auf Basis der DFA sprechen für die Notwendigkeit der Weiterentwicklung über die derzeitige Projektlaufzeit von DARIAH-DE hinaus.

4.2. Data Modeling Environment (DME)

Das DME wurde im Verlauf dieses Jahres als DFA Komponente eingeführt und löst die SR ab. Das vorliegende Dokument ist das erste, welches das DME als Komponente explizit nennt und die Weiterentwicklung der SR zu einem umfangreichen Werkzeug zur Datenmodellierung und -integration markiert.
Ursprünglich war für Schemata und Crosswalks eine einfache Registrierungs- und Verwaltungskomponente geplant (vgl. z. B. Henrich/Gradl, 2013). Bereits im Verlauf der ersten Förderphase von DARIAH-DE hat sich abgezeichnet, dass ein derartig einfaches Konzept für eine forschungsnahe Unterstützung der Beschreibung von Datenstrukturen und deren Zusammenhänge nicht ausreicht. Basierend auf dem Konzept domänenspezifischer Sprachen wurden neuartige Methoden a) zur Spezifikation von Daten und b) deren Transformation entwickelt, die zwischenzeitlich erprobt und publiziert wurden (vgl. z. B. Gradl/Henrich, 2016a; Gradl/Henrich, 2016c).
Auch weil eine ausführliche Nutzerdokumentation des DME und der zu Grunde liegenden Konzepte bislang weitgehend fehlt, bieten die folgenden Abschnitte etwas umfassenderer Erklärungen zu diesen Themen.

4.2.1. Definition und Transformation von Daten

Datentransformation und -integration sind keine neuen Themen, haben in der wissenschaftlichen Forschung jedoch nie an Aktualität verloren. Ausgehend von Datenbanksystemen und der Notwendigkeit, integrative Sichten über unterschiedlich strukturierte Daten zu erstellen, werden für Anwendungsfälle aufgrund individueller Rahmenbedingungen oft neue Methoden entwickelt bzw. individuell angepasst. Gerade in der Welt

[22] Bspw. Die Verbundsuche des Forschungsverbunds Marbach Weimar Wolfenbüttel (MWW)

von XML hat sich XSL Transformation[23] als wichtiges und für eine Vielzahl von Anwendungsfällen gut geeignetes Mittel durchgesetzt. Unterschiedliche Transformationssprachen, wie die Query View Transformation (QVT)[24] der Meta Object Facilities (MOF) sind dagegen *entkoppelt von konkreten Formaten*, weisen jedoch einen hohen Grad an *Komplexität* auf und setzen das Vorliegen fertig vorverarbeiteter Ausgangsdaten voraus. Eine wesentliche Eigenschaft von Forschungsdaten – gerade im Kontext der Kultur- und Geisteswissenschaften – besteht jedoch darin, dass für deren Interpretation häufig die Anwendung von Hintergrundwissen z. B. über die beinhaltende Kollektion erforderlich ist. Möglichkeiten zur Explikation v. a. komplexeren Hintergrundwissens ist in bestehenden Transformationssprachen und -systeme nicht gegeben.

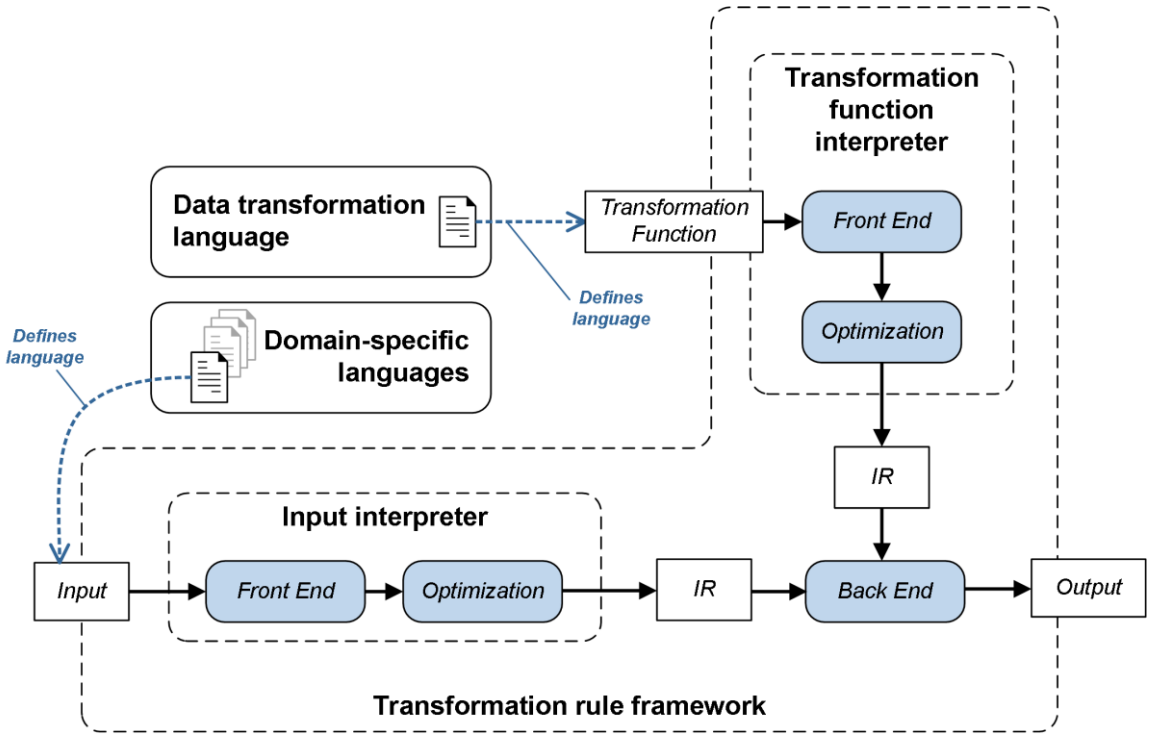


Abbildung 6: Übersicht Transformationsframework (Gradl, 2014)

Die Besonderheit des DME Konzepts besteht in *der Abspaltung der Datendefinition von der eigentlichen Transformation*. Abbildung 6 verdeutlicht das zu Grunde liegende Funktionsprinzip: Daten (Input) werden grammatikalisch verarbeitet und stehen dadurch in einer interpretierten Repräsentation (intermediate representation, IR) zur Verfügung. Ebenso werden auch anzuwendende Transformationsanweisungen zunächst interpretiert und im Hinblick auf deren Anwendung optimiert. Das so genannte

[23] https://www.w3.org/TR/xslt/
[24] http://www.omg.org/cgi-bin/doc?ptc/2007-07-07

Back End[25] führt beide Repräsentationen zusammen und führt die Transformation schließlich durch.[26]

Die Definition der Daten vor der eigentlichen Transformation ermöglicht in vielen Fällen eine Explikation von Hintergrundwissen im Rahmen des Datenmodells und insbesondere auch die deutliche Vereinfachung von Transformationsregeln. Die Definition der Daten basiert methodisch auf dem Konzept domänenspezifischer Sprachen (Gradl/Henrich, 2016a): fachwissenschaftliche Expertinnen und Experten spezifizieren Regeln, die festlegen, wie Elementinhalte in konkreten Kontexten zu interpretieren sind. Eine Verdeutlichung dieses Konzepts anhand von Beispielen folgt in Abschnitt 4.3.

4.2.2. Erstellungs- und Verwendungskontext

Eine weitere wesentliche Idee des DME und des zu Grunde liegenden Konzepts besteht in der Unterscheidung des Erstellungs- und Verwendungskontexts von Daten. Beide können im Rahmen des DME expliziert und modelliert werden, ihre Unterscheidung beeinflusst die Nachnutzbarkeit modellierten Wissens.

Als **Erstellungskontext** können *Rahmenbedingungen bei der Generierung und Weiterverarbeitung von Forschungsdaten in ihrer originären Umgebung* zusammengefasst werden. Diese sind abhängig von der Domäne und Kollektion in denen sie entstehen. So liegen beispielsweise oft Regeln vor, wie Eigennamen oder Listen von Eigenschaften abzubilden sind. Weitere Bestandteile des Erstellungskontexts sind häufig Informationen der Kollektion oder Domäne, die für sämtliche Inhalte gelten und aus diesem Grund nicht in den Daten selbst auftauchen. In ihrem ursprünglichen Kontext betrachtet können die Daten durch Anwendung impliziter Kontextinformation problemlos interpretiert werden. Sobald Daten aus diesem Kontext entnommen und insbesondere mit weiteren Daten zusammengeführt werden, wird deren Interpretation deutlich erschwert. Die Modellierung des Erstellungskontexts von Daten verbessert deren Qualität – unabhängig von der späteren Verwendung der Daten. Sie erfolgt möglichst am Datenmodell selbst, nicht im Rahmen von Mappings.

---

[25] Das Konzept und die Begriffe entstammen dem Themenfeld des Compiler Engineering – für weitere Recherchen kann bspw. das Lehrbuch Cooper/Torczon (2011) herangezogen werden.

[26] Für eine vertiefende technische Betrachtung vgl. Gradl/Henrich, 2016a oder Gradl, 2014

Als ***Verwendungskontext*** können die *Rahmenbedingungen der Verwendung bzw. Interpretation* von Daten zusammengefasst werden. Eine direkte Verwendung findet zumeist im Erstellungskontext der Daten statt, so dass sich der erste Verwendungskontext ggf. nicht von diesem unterscheidet. Weitere Rahmenbedingungen treten jedoch häufig bereits innerhalb der originären Umgebung auf – beispielsweise dann, wenn Daten über erstellungsfremde Kanäle angeboten werden sollen. Sollen Daten beispielsweise über eine Suchschnittstelle zugänglich sein, so bietet sich ggf. eine gezielte Bereinigung und Anreicherung von Daten im Hinblick auf diese Art der Verwendung an. Die Modellierung des Verwendungskontexts erfolgt mit Hilfe des Mappings eines Datenmodells auf ein gewünschtes Zielmodell. Elemente innerhalb des Mappings werden durch Abbildungen von Konzepten (*concept mappings*) gebildet und können einfache, atomare (1:1) Wertkorrespondenzen oder auch komplexere (M:N) Verbindungen einnehmen.

4.3. Modellierungsbeispiele

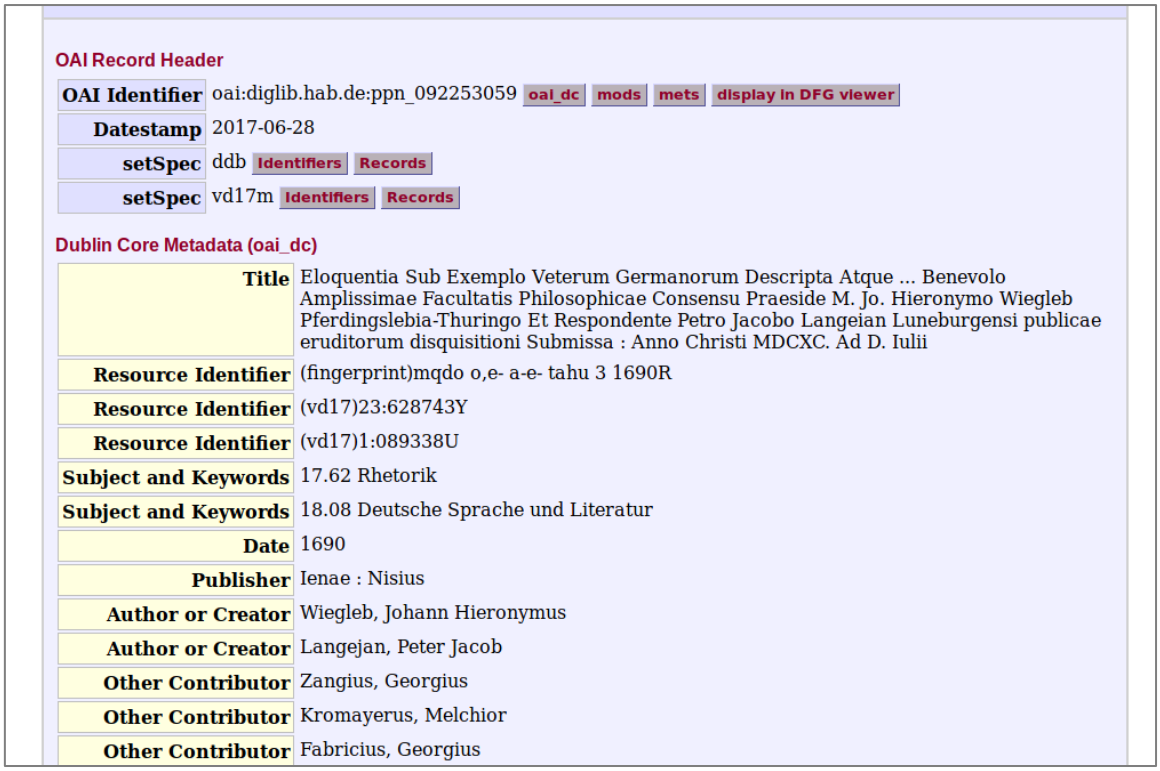
Die technischen Grundlagen und Kernideen des DME benötigen gerade wegen ihres Ursprungs in der theoretischen Informatik sicherlich einer grundlegenderen Einführung, als dies im Rahmen eines solchen Meilensteinberichts erfolgen kann und sollte. Einerseits sind die technischen Hintergründe insbesondere für geisteswissenschaftliche Leserinnen und Leser eher von untergeordnetem Interesse, andererseits wurden und werden diese in verschiedenen technischen Publikationen vertieft und können bei Bedarf dort nachgelesen werden. In diesem Abschnitt sollen einführende Modellierungsbeispiele dabei helfen, die technischen Konzepte praktisch einzuordnen. Die Beispiele wurden so gewählt, dass diese auf weitere Anwendungsfälle übertragen werden können und insbesondere auch der Nutzen einer Unterscheidung von Datendefinition und Datentransformation erkennbar ist. Sie sind nicht als vollständige und unbedingt korrekte Abbildung eines fachwissenschaftlichen Kontexts zu verstehen.

4.3.1. Einfaches Dublin Core

Für ein erstes Modellierungsbeispiel werden Daten von der OAI-PMH Schnittstelle der Herzog August Bibliothek Wolfenbüttel[27] (Abbildung 7) bezogen. Diese Daten sollen

[27] http://oai.hab.de/?verb=ListRecords&metadataPrefix=oai_dc

mit Hilfe des DME interpretiert werden können. Darüber hinaus sollen Daten durch deren grammatikalische Definition angereichert werden. Das Beispiel ist zu Gunsten der Nachvollziehbarkeit bewusst so gewählt, dass für die Formulierung der Grammatiken und die Abbildung des Datenmodells kein spezifisches Domänenwissen zu der beinhaltenden Kollektion oder der HAB erforderlich ist.



OAI Record Header

OAI Identifier	oai:diglib.hab.de:ppn_092253059 oai_dc mods mets display in DFG viewer
Datestamp	2017-06-28
setSpec	ddb Identifiers Records
setSpec	vd17m Identifiers Records

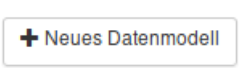
Dublin Core Metadata (oai_dc)

Title	Eloquentia Sub Exemplo Veterum Germanorum Descripta Atque ... Benevolo Amplissimae Facultatis Philosophicae Consensu Praeside M. Jo. Hieronymo Wiegleb Pferdingslebia-Thuringo Et Respondente Petro Jacobo Langeian Luneburgensi publicae eruditorum disquisitioni Submissa : Anno Christi MDCXC. Ad D. Iulii
Resource Identifier	(fingerprint)mqdo o,e- a-e- tahu 3 1690R
Resource Identifier	(vd17)23:628743Y
Resource Identifier	(vd17)1:089338U
Subject and Keywords	17.62 Rhetorik
Subject and Keywords	18.08 Deutsche Sprache und Literatur
Date	1690
Publisher	Ienae : Nisius
Author or Creator	Wiegleb, Johann Hieronymus
Author or Creator	Langejan, Peter Jacob
Other Contributor	Zangius, Georgius
Other Contributor	Kromayerus, Melchior
Other Contributor	Fabricius, Georgius

Abbildung 7: Beispielantwort der OAI-PMH Schnittstelle der HAB Wolfenbüttel

Um Daten mit Hilfe der DME verarbeiten zu können, ist eine Modellierung ihrer Struktur erforderlich. Im Fall von Dublin Core stehen entsprechende Modelle bereits zur Verfügung – für eine verbesserte Nachvollziehbarkeit der Beispiele wird hier ein dediziertes Datenmodell angelegt. Das gesamte Modell steht auch in der DFAtest Instanz des DME zur Verfügung.[28]

Notwendige Schritte für den Import eines XML Schemas als Datenmodell sind:

1. Aufruf von https://dfatest.de.dariah.eu/dme oder https://dme.de.dariah.eu/dme
2. Anmeldung als Benutzer nach Auswahl von „Login“
3. 

→ Bezeichnung eingeben → 

[28] Beispielmodell: https://dfatest.de.dariah.eu/dme/model/editor/59fc37bdd8ad376894ae3186/

4. 

5. Auswahl der zu importierenden Datei, hier: *oai_dc.xsd*[29]

Nach einem erfolgten Import des Schemas wird das Elementmodell vergleichbar zu Abbildung 8 dargestellt. Um auch die Ergebnisse einer Beispieltransformation von Daten (in der Abbildung links) anzuzeigen, kann nach Betätigung der Schaltfläche  eine geeignete Datei[30] hochgeladen werden.

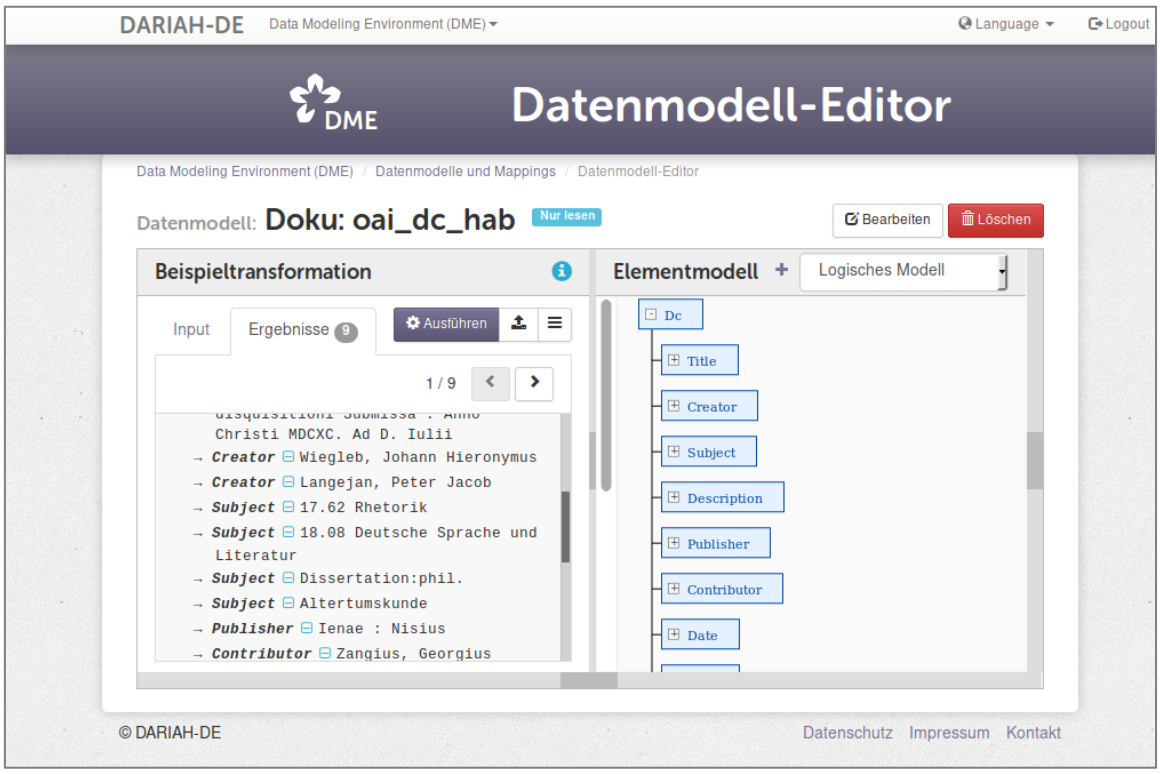


Abbildung 8: Importiertes Dublin Core Schema

Als einfache Beispiele für die weitere Modellierung werden im Folgenden die Elemente *Creator*, *Subject* und *Publisher* betrachtet. Die Interaktionsfolge für die Verfeinerung von Elementen ist für alle Beispiele identisch:

1. Unter dem Element wird eine *Grammatik* angelegt.

[29] http://www.openarchives.org/OAI/2.0/oai_dc.xsd
[30] Im konkreten Beispiel: eine Antwort des OAI-PMH Service der HAB Wolfenbüttel

2. Unter der neuen Grammatik wird eine *Transformationsfunktion* angelegt.
3. Unter der Funktion wird schließlich die zu *produzierende Elementhierarchie* modelliert.

In der DME ergibt sich je modelliertem Element in etwa die in Abbildung 9 dargestellte Ansicht: Im Ursprungsformat enthaltene Strukturelemente sind *blau*, Grammatik und Transformationsfunktion *orange* und zu produzierende Elemente *violett* dargestellt.

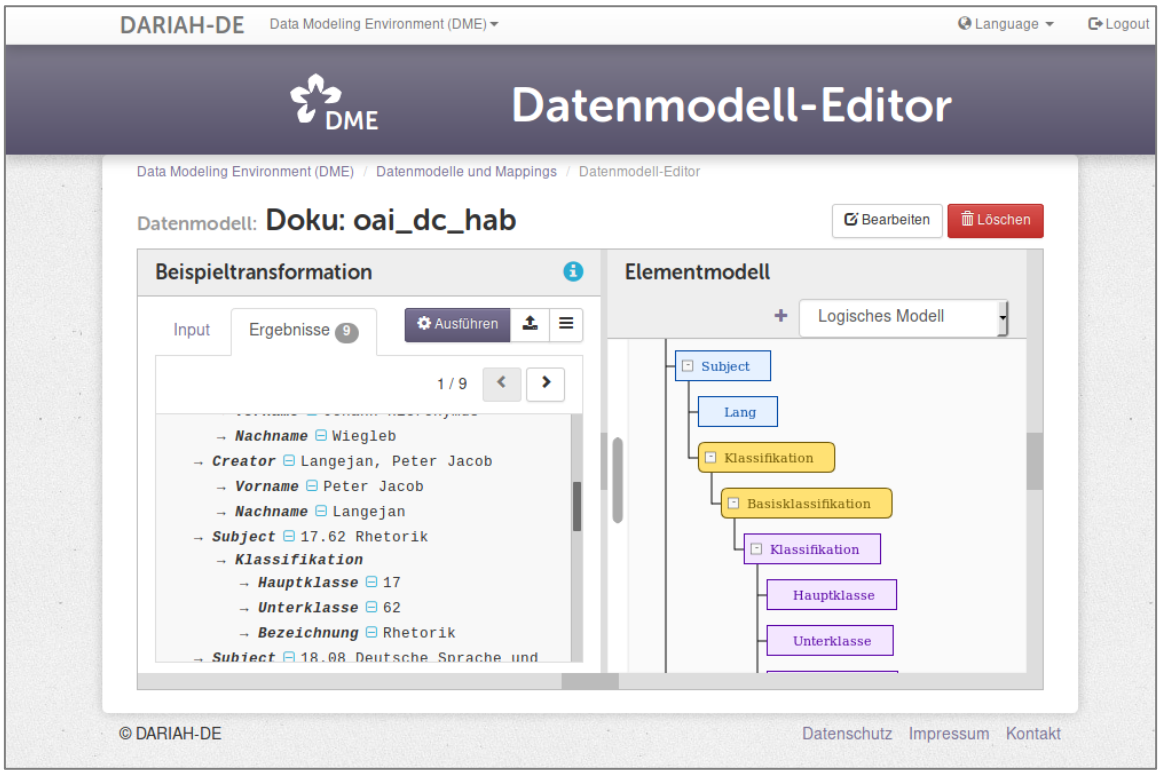


Abbildung 9: Verfeinertes oai_dc_hab Datenmodell

Für die Interpretation des Namens in *Creator* kann eine einfache Grammatik wie in Abbildung 10 dargestellt formuliert werden. Grammatiken in der DME folgen der Erweiterten Backus-Naur-Form (EBNF) und formulieren Produktionsregeln einer Sprache. Unterschieden werden lexikalische Regeln zur Bildung von Tokens – den Worten der Sprache – und syntaktische Regeln.

Für ***Creator*** gibt es nur eine Wortart STRING, die dadurch definiert ist, dass sie mindestens ein Zeichen (+) aufweist, welches nicht dem Komma entspricht. Anhand dieser einen Regel produziert die Grammatik für den Beispielnamen *Wiegleb, Johann Hieronymus* die drei Worte *Wiegleb*, *(Komma)* und *Johann Hieronymus*. Die Parser-Regeln sind dann:

- Ein *name* besteht aus einem *lname* gefolgt von *(Komma)* und einem *fname*
- *fname* und *lname* sind genau ein beliebiges Wort *STRING*

„Wiegleb, Johann Hieronymus“

```
name : lname ', ' fname;
lname : STRING;
fname : STRING;

STRING : ~(',')+;
```

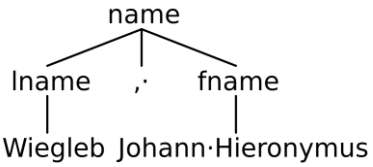


Abbildung 10: Grammatik für Creator und resultierender Syntaxbaum

Durch eine derartige Definition von Daten und die daraus resultierende Verarbeitung können sehr einfache Transformationsregeln formuliert werden, die lediglich in einer Wertzuweisung der Form *Outputelement* = *@regelname* bestehen:

```
Vorname = @fname;
Nachname = @lname;
```

Bereits in diesem einfachen Beispiel ist der Vorteil der Trennung von Definition und Transformation ersichtlich: die Sprache der Daten ist als intrinsische Eigenschaft unabhängig von der eigentlichen Transformation und ermöglicht die Reproduktion der Regeln, die bei der Erstellung der Daten implizit angewendet wurden. Würde beispielsweise neben oder anstelle der individuellen Speicherung von Vor- und Nachname eine Darstellung der Form (Vorname Nachname) gewünscht, müsste nur die Transformationsregel, nicht jedoch die Grammatik angepasst werden:

```
Name = CONCAT(@fname, „ “, @lname);
```

Die CONCAT Funktion dient der Verknüpfung von Elementen und fügt hier fname, ein Leerzeichen und lname aneinander, um das neue Element Name zu erzeugen.

Die Verarbeitung des ***Publisher*** Elements (Abbildung 11) basiert auf vergleichbaren Regeln und benötigt nach Anwendung der Grammatik ebenfalls nur einfache Wertzuweisungen. Der Ort wird den Daten entsprechend als optional definiert.

„Lenae : Nisius“

```
publisher : (ort ':')? herausgeber;
herausgeber : STRING;
ort : STRING;

STRING : ~(':')+;
```

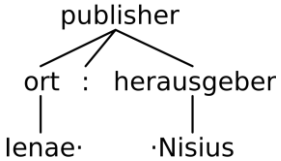


Abbildung 11: Grammatik für Publisher und resultierender Syntaxbaum

Die grammatikalische Definition des ***Subject*** Elements bedingt die Abbildung der Basisklassifikation und ist etwas komplexer als die bisherigen Beispiele. Hier werden vier Regeln zur Produktion von Worten spezifiziert. Eine *ZAHL* ist eine Folge von Ziffern, ein *STRING* kann aus beliebigen Zeichen außer Ziffern dem Punkt und dem Leerzeichen bestehen. Der Punkt (*DOT*) wird aufgrund seiner Eigenschaft als Trenner von Hauptklasse und Unterklasse als eigene Wortart definiert, ebenso wie das Leerzeichen (*WS*), welches die Unterklasse von der Bezeichnung trennt. Da in der Bezeichnung neben *STRING*s auch Zahlen, Leerzeichen und Punkte auftauchen können, wird diese mit Hilfe von Optionen (|) und eines geeigneten Multiplikators (+) definiert.

```
klassifikation : hauptklasse
         DOT unterklasse
         WS bezeichnung;
hauptklasse : ZAHL;
unterklasse : ZAHL;
bezeichnung : (STRING | ZAHL
               | WS | DOT)+;
ZAHL : [0-9]+;
STRING : ~[0-9. ]+;
DOT : '.';
WS : ' ';
```

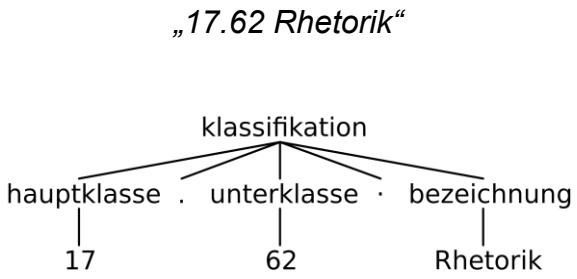


Abbildung 12: Grammatik für Subject und resultierender Syntaxbaum

4.3.2. Archäologischer Use-Case

Betrachten wir den Auszug eines Datensatzes aus dem *Pergamon* Projekt des Deutschen Archäologischen Instituts (DAI) in Berlin. Das Projekt diente bereits als Beispiel-Anwendungsfall für den im August 2017 veröffentlichten DARIAH-DE Report 4.2.2 *Publikation der Rückmeldungen und Beobachtungen zu den integrierenden Use-Cases*[31].

[31] https://wiki.de.dariah.eu/download/attachments/14651583/R4.2.2.pdf?version=1&modificationDate=1502967300654&api=v2

Abbildung 13 zeigt einen kurzen Auszug eines exportierten Datensatzes `<ROW>`, für den durch eine entsprechende Kontextmodellierung eine semantische Anreicherung erreicht werden kann. Das Beispielmodell ist in der DFAtest Instanz hinterlegt.[32]

```
<FMPDSORESULT xmlns="http://www.filemaker.com/fmpdsoresult">
    <ERRORCODE>0</ERRORCODE>
    <DATABASE>DAIKeramik.fmp12</DATABASE>
    <LAYOUT/>
    <ROW MODID="84" RECORDID="15755">
        <PS_KeramikID>36674</PS_KeramikID>
        ...
        <funktionale_Verwendung>Tafelkeramik</funktionale_Verwendung>
        <Gattung/>
        <Grobdatierung>hellenistisch</Grobdatierung>
        <Grösse_Breite>0,5-0,6</Grösse_Breite>
        ...
        <Provenienz>Pergamon</Provenienz>
        ...
    </ROW>
</FMPDSORESULT>
```

Abbildung 13: Auszug eines Datensatzes aus dem Pergamon-Projekt

Das Feld `<Grobdatierung>` beinhaltet unstrukturierten Inhalt mit einer groben zeitlichen Angabe. Im konkreten Beispiel atomar, existieren auch Angaben wie

`hellenistisch-kaiserzeitlich`,

deren Interpretation ohne kontextuelles Hintergrundwissen nicht ohne Ambiguitäten erfolgen kann. So ist unklar, ob die Angabe beispielsweise nur einen kurzen Zeitraum des Übergangs zwischen den beiden Zeitangaben adressiert. Tatsächlich entspricht die für den Anwendungsfall korrekte Interpretation einem von-bis Muster, konkret also „von hellenistisch bis kaiserzeitlich“.

Angaben im Feld `<Grobdatierung>` sind demnach nicht völlig unstrukturiert, sondern können mit Hilfe einer einfachen Grammatik definiert werden.

„hellenistisch-kaiserzeitlich“

```
epochs : epoch ('-' epoch)*;
epoch : STRING;

STRING : ~('-')+;
```

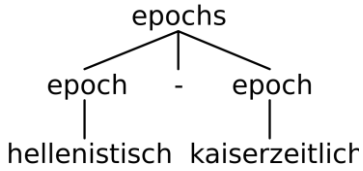


Abbildung 14: Grammatik für `<Grobdatierung>` und Syntaxbaum

[32] https://dfatest.de.dariah.eu/dme/model/editor/5a001287d8ad376894ae31c2/

Durch den Aufruf der *Chronontology* Schnittstelle des DAI können Datierungsangaben angefragt und aufgelöst werden. Durch die Vorverarbeitung der Daten lautet der Befehl zum Aufruf der Schnittstelle:

```
Chronontology = DAI::CHRONONTOLOGY::QUERY(@epoch);
```

Der Bildschirmausschnitt in Abbildung 15 zeigt die erweiterte Struktur des Pergamon Datenmodells im rechten, die Ergebnisse der Verarbeitung eines Beispieldatensatzes im linken Bereich des Editors. Unterhalb von Grobdatierung finden sich in beiden Bereichen die jeweils angereicherten Elemente.

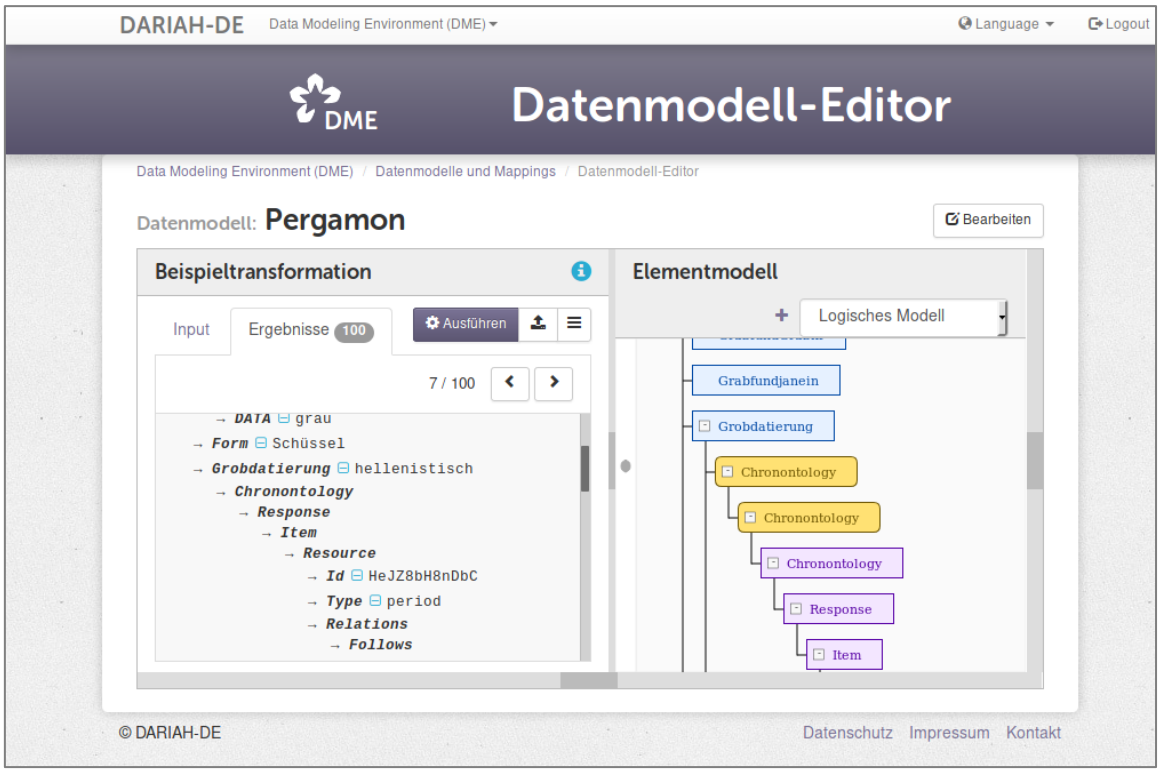


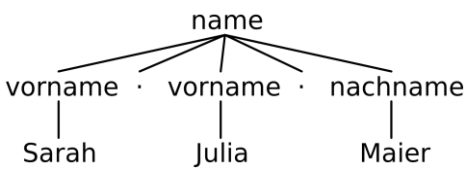
Abbildung 15: Modellierungsbeispiel Erfassungskontext „Pergamon“

Weitere Beispiele für die Definition des Erstellungskontexts der Pergamon Daten finden sich in Abbildung 16:

```
name : (vorname ' ')+ nachname;
vorname: STRING;
nachname: STRING;

STRING: ~(' ')+;
```

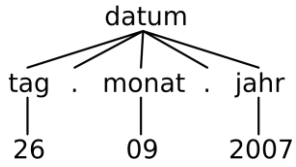
„Sarah Julia Maier“



```
datum : tag '.' monat '.' jahr;
tag : DIGIT;
monat : DIGIT;
jahr : DIGIT;

DIGIT: [0-9]+;
```

„26.09.2007“




```
color : hue ' ' value '/' chroma;

hue : number ' ' colorcode;
value: number;
chroma: number;

number: DIGIT+ (',' DIGIT+)?;
colorcode: LETTER LETTER?;

DIGIT: [0-9];
LETTER: [RYGBP];
```

„5 YR 6/6“

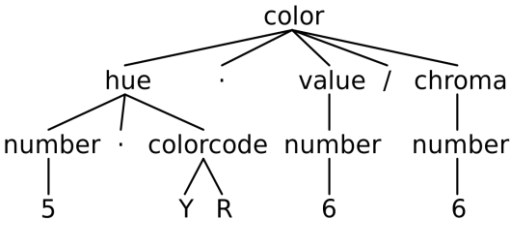


Abbildung 16: Weitere Beispiel-Grammatiken und Syntaxbäume

Verwendungskontext Geobrowser

Abbildung 17 zeigt den Ausschnitt eines Mappings zwischen dem Pergamon Datenmodell und einer vereinfachten Variante der Keyhole Markup Language (KML). Eine Wertkorrespondenz liegt hier bei der Abbildung von {Provenienz} → {Address} vor: es sind keine weiteren Regeln formuliert, wodurch Inhalte unverändert in das Zielmodell übernommen werden. Für die Abbildung {Latitude, Longitude} → {Coordinates} ist die Anwendung einer Transformationsregel erforderlich:

```
[@Latitude!=null]
  Coordinates = CONCAT(@Latitude, ",", @Longitude)
[endif]
```

Die Regel verbindet die Werte – getrennt durch ein Komma – zu einem String. Aus Latitude 27.183953, Longitude 39.132126 entsteht „27.183953,39.132126“. Die Anweisung beinhaltet auch eine konditionale Einschränkung, welche nur dann eine Ausgabe im Zielmodell zulässt, wenn ein Wert für Latitude (implizit dann auch für Longitude vorliegt).

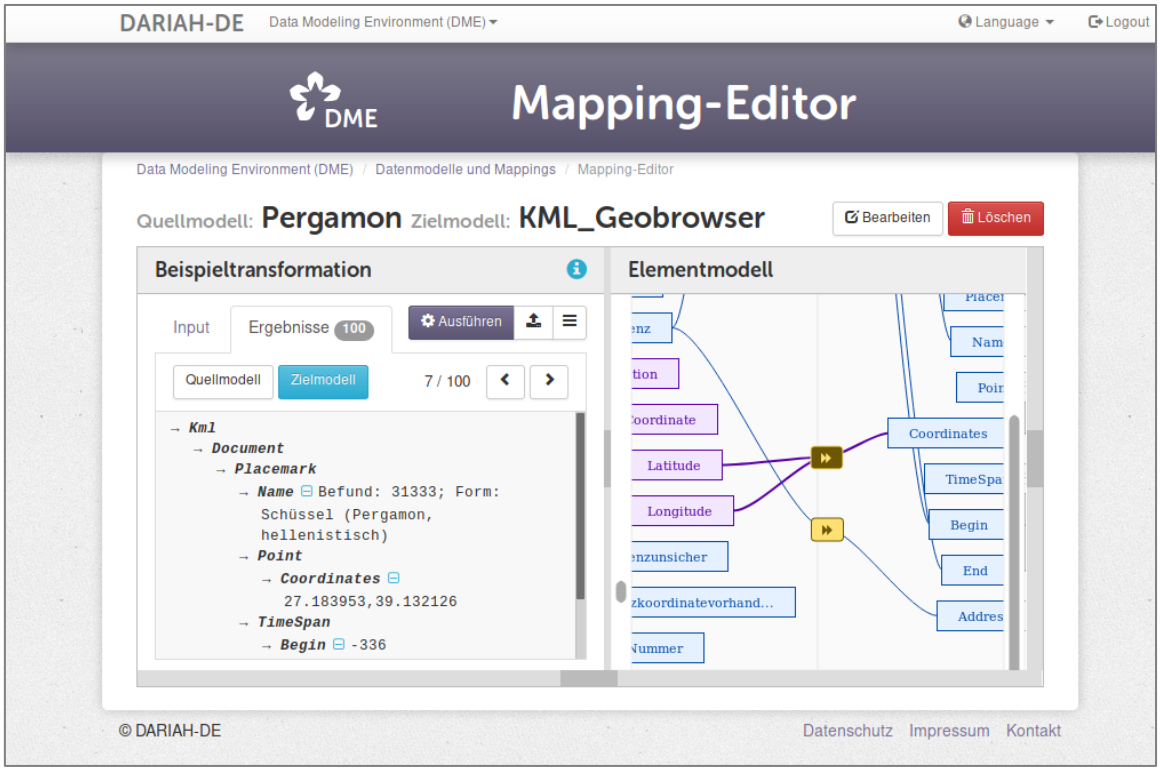


Abbildung 17: Modellierungsbeispiel „Pergamon“ in neuem Kontext KML

5. Ausblick

Sämtliche Komponenten der DFA haben einen Entwicklungsstand erreicht, der eine Anwendbarkeit für Forschungsdaten ermöglicht. Mit Hilfe der CR werden Sammlungen verzeichnet und beschrieben, die eine Relevanz für Forscherinnen und Forscher aufweisen. Das DME wird in unterschiedlichen Kontexten eingesetzt. Neben den Installationen auf Produktiv- und Testsystemen von DARIAH-DE existieren weitere Instanzen, die für Machbarkeitsstudien und die Vorbereitung weiterführender Projektanträge eingesetzt werden. Das DARIAH-DE Repository wird parallel zu diesem Bericht als Produktivsystem installiert und steht Forscherinnen und Forschern für die Publikation von Forschungsdaten zur Verfügung.

Wenngleich bislang eine gewisse Stabilität und Nutzbarkeit der Komponenten erreicht werden konnte, so werden durch jeden unterstützten Anwendungsfall geisteswissenschaftlicher Forschung neue Anforderungen an die Komponenten gestellt. So zeichnet sich für die weitere Entwicklung innerhalb der derzeitigen Projektphase von DARIAH-DE zwar eine Erreichbarkeit gesteckter Ziele ab – das Interesse aus der fachlichen Community verdeutlich jedoch auch die Notwendigkeit der Weiterentwicklung der Komponenten – z. B. weitere Erweiterungsbibliotheken des GTF.

6. Glossar

CR	Collection Registry
DFA	Datenföderationsarchitektur
DME	Data Modeling Environment
SR	Schema Registry

7. Literaturverzeichnis

Die Verfügbarkeit im Bericht angegebener Links wurde zuletzt am 09.11.2017 geprüft.

Cooper, Keith D./Torczon, Linda (2012): Engineering a compiler. 2. ed. Amsterdam: Elsevier Morgan Kaufmann

Gradl, Tobias (2014): Concept and implementation of a rule framework to dynamically transform data and queries for heterogeneous collections. Master thesis. Media Informatics Group. Bamberg. URL: https://doi.org/10.5281/zenodo.804889

Gradl, Tobias/Henrich, Andreas (2016a): „Data Integration for the Arts and Humanities: A Language Theoretical Concept“. In: Fuhr, Norbert et al. (Hg.): *Research and Advanced Technology for Digital Libraries: 20th International Conference on Theory and Practice of Digital Libraries, TPDL 2016, Hannover, Germany, September 5-9, 2016, Proceedings.* Cham: Springer International Publishing, S. 281–293

Gradl, Tobias/Henrich, Andreas (2016b): „Die DARIAH-DE-Föderationsarchitektur – Datenintegration im Spannungsfeld forschungsspezifischer und domänenübergreifender Anforderungen“. In: Bibliothek Forschung und Praxis 40, H. 2

Gradl, Tobias/Henrich, Andreas (2016c): „Extending Data Models by Declaratively Specifying Contextual Knowledge“. In: Sablatnig, Robert/Hassan, Tamir (Hg.): DocEng '16 Proceedings of the 2016 ACM Symposium on Document Engineering. New York, NY, USA: ACM: ACM, S. 123–126

Gradl, Tobias/Henrich, Andreas/Plutte, Christoph (2015): „Heterogene Daten in den Digital Humanities: Eine Architektur zur forschungsorientierten Förderation von Kollektionen“. In: Baum, Constanze/Stäcker, Thomas (Hg.): *Grenzen und Möglichkeiten der Digital Humanities*Zeitschrift für digitale Geisteswissenschaften. 2015, H. 1. URL: http://zfdg.de/sb001_020 [Stand: 13. Januar 2016]

Henrich, Andreas/Gradl, Tobias (2013): „DARIAH(-DE): Digital Research Infrastructure for the Arts and Humanities — Concepts and Perspectives“. In: International Journal of Humanities and Arts Computing 7, H. supplement, S. 47–58

8. Abbildungsverzeichnis