Florian Thiery B.Sc. (900841)

Semantic Web und Linked Data: Generierung von Interoperabilität in archäologischen Fachdaten am Beispiel römischer Töpferstempel

Masterarbeit

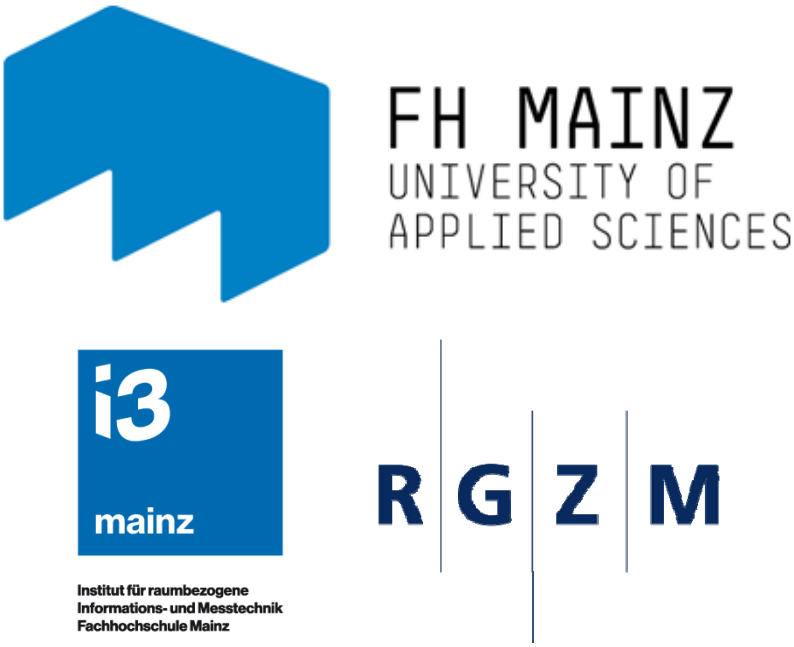
zur Erlangung des akademischen Grades Master of Science im Studiengang Geoinformatik und Vermessung

Fachhochschule Mainz
Fachbereich Technik
Lehreinheit Geoinformatik und Vermessung

Betreuer: Prof. Dr. phil. Kai-Christian Bruhn
Bearbeitungszeitraum: 13. Juni 2013 – 11. Dezember 2013

Standnummer: KM0042

Mainz im Dezember 2013



Semantic Web und Linked Data: Generierung von Interoperabilität in archäologischen Fachdaten am Beispiel römischer Töpferstempel

Florian Thiery B.Sc.

Kurzfassung

Gegenstand dieser hier vorgestellten Masterarbeit ist Verwendung aktueller Technologien interoperabler Datenhaltung, insbesondere das Konzept der Linked Open Data (LOD) und der semantischen Modellierung, zur Verdeutlichung ihres Potentials in archäologischen Informationen am Beispiel von Terra Sigillata-Fundorten, -Töpfern und -Keramikfragmenten. Die Arbeit zeigt eine Migration von Daten sowie die Möglichkeiten und die Problematik der Modellierung der Attribute und Beziehungen mit Hilfe bestehender LOD-Konzepte und kontrollierter Vokabularien, sowie eigene Ansätze zur Lösung. Diese Daten werden mittels ReST-Schnittstelle zur Verfügung gestellt. Ein Schwerpunkt wird auf die Verlinkung zu anderen bereits bestehenden Projekten gelegt, wodurch eine Vielzahl weiterer archäologischer und historischer Informationen z.B. über das Pelagios Projekt eingebunden werden. Zudem wird das Potential der Verlinkung und Abfrage von heterogenen Informationen zwischen Töpfern, Fragmenten und Orten deren relativ chronologische Beziehungen über LOD mit einer webbasierten Schnittstelle aufgezeigt.

Schlagwörter: Semantic Web, Linked Data, ReST, Chronologie, Terra Sigillata

Abstract

The subject matter of this master thesis is using current technologies in interoperable data management, in particular the illustration of the potential of Linked Open Data (LOD) and semantic modelling in archaeological information, as used on samian ware places and their corresponding potters and ceramic fragments. The thesis demonstrates a migration of data as well as possibilities and problems of modelling attributes and relationships using existing LOD concepts and controlled vocabularies as well as novel self-developed approaches to the solution. These data are provided by a ReST interface. One focus is linking to other existing projects, creating associations to other archaeological and historical information, for example the Pelagios project. Moreover, a web-based interface shows the potential of linking and retrieval of heterogeneous information among pottery, fragments and places and their relative chronological relationships via LOD.

keywords: semantic web, linked data, REST, chronology, samian ware

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbständig angefertigt habe. Es wurden nur die in der Arbeit ausdrücklich benannten Quellen und Hilfsmittel benutzt. Wörtlich oder sinngemäß übernommenes Gedankengut habe ich als solches kenntlich gemacht.

Mainz, den 11. Dezember 2013

Florian Thiery B.Sc.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

GeInArFa	Generierung von Interoperabilität in archäologischen Fachdaten Kurzform der hier beschriebenen Masterarbeit
i3mainz	Institut für Raumbezogene Information- und Messtechnik
RGZM	Römisch Germanisches Zentralmuseum Mainz
LOD	Linked Open Data
RDF	Ressource Description Framework
SPARQL	SPARQL Protocol And RDF Query Language
OWL	Web Ontology Language
ReST	Representational State Transfer (auch REST)
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
OGC	Open Geospatial Consortium
WMS	Web Map Service
WFS	Web Feature Service
SLD	Styled Layer Descriptor
GML	Geography Markup Language
EPSG	European Petroleum Survey Group Geodesy
XML	Extensible Markup Language
OA	Open Annotation
LSWE	Linked Samian Ware Explorer

Vorwort

"It's difficult to imagine the power that you're going to have when so many different sorts of data are available."

Sir Timothy John Berners-Lee[1]

Erfinder der Hypertext Markup Language und Begründer des World Wide Web

Die erste Seite dieser Masterarbeit möchte ich nutzen, um all denen zu danken, die mich bei der Anfertigung dieser Arbeit in den letzten Wochen und Monaten unterstützt haben.

Mein größter Dank gilt Herrn Prof. Dr. Bruhn, der es mir ermöglicht hat, meine Masterarbeit am Institut für Raumbezogene Informations- und Messtechnik der Fachhochschule Mainz (i3mainz) in Zusammenarbeit mit dem Römisch-Germanischen Zentralmuseum Mainz (RGZM) zu schreiben. Für die sehr gute Betreuung und tatkräftige Unterstützung zu jeder Tag- und Nachtzeit sowie das meinem Thema entgegengebrachte Interesse bedanke ich mich vielmals. Ein weiterer großer Dank ergeht ebenfalls den Mitarbeitern des RGZM Guido Heinz und Allard Mees, die mich sowohl in technischen, wie vor allem in archäologischen Fragestellungen unterstützt haben. Ein großer Dank ergeht auch an den ‚technischen Support': Nikolai Bock, Thomas Engel und Georg Paternoster, wie auch an sonstige Mitarbeiter diverser Institute wie Torsten Schrade, Martin Unold, Michael Haft und Tobias Kohr.

Meiner Mutter Susanne, meinem Vater Peter wie auch allen anderen Familienangehörigen kann ich für die lebenslange Unterstützung, besonders während des Studiums und die mir entgegengebrachte Liebe und Geduld nicht genug danken. Bedanken möchte ich mich zudem bei all meinen Freunden, die ihre Ansichten über die Ergebnisse meiner Arbeit mit mir geteilt und diskutiert haben.

Florian Thiery, Mainz den 11. Dezember 2013

[1] vgl. http://www.w3.org/People/Berners-Lee

1 Motivation und Aufgabenstellung

Das Zeitalter des römischen Imperiums ist insbesondere für die Einführung einer rot schimmernden Keramik bekannt, der sogenannten Terra Sigillata (TS) oder auch Samian Ware. Es ist ein gehobenes Tafelgeschirr, das in großen Mengen in Handwerksbetrieben (Manufakturen) hergestellt und meist mit Stempeln (Töpfermatrizen) versehen wurde. Die so erstellte Ware wurde im gesamten römischen Reichen in verschiedenen Variationen[2] verhandelt. TS wurde bereits in der zweiten Hälfte des 1. Jahrhunderts v. Chr. in Oberitalien, besonders im Hauptproduktionsort Arezzo entwickelt[3]. „Während der beiden letzten Jahrzehnte des 1. Jahrhunderts v. Chr. gründeten einige in Arezzo arbeitende Großproduzenten Filialbetriebe in Pisa und in Gallien. Diese italienische Manufaktur am oberen Arno belieferte das gesamte Mittelmeergebiet sowie die nordwestlichen Provinzen des Römischen Reiches. Ab etwa 20 v. Chr. ist der Beginn der umfangreichen Produktion im ca. 150km entfernten Pisa an der Arnomündung wahrnehmbar. […] Ab 15/20 n. Chr. explodierte die Großmanufaktur in La Graufesenque. Im Laufe des 1. Jahrhunderts n. Chr. verdrängte sie [die südgallische Terra Sigillata] die italienische Sigillata aus dem Verbreitungsraum von Britannien bis zur mittleren Donau vom Markt und nahm eine Zeit lang in den nordwestlichen Provinzen eine Monopolstellung ein. [MEES, 2011]“. Bereits vor hundert Jahren wurde durch Dragendorff et al.[4] auf einen Zusammenhang zwischen Handelswegen und Relief-Sigillaten hingewiesen. Mees erforscht diesen Zusammenhang mit Hilfe von quantifiziertem Datenmaterial. „Zur Beantwortung der Fragen, warum die arretinischen Großbetriebe Zweigmanufakturen in Gallien gründeten […] wurden die ca. 20 000 Namenstempel der italienischen und sudgallischen Sigillata Töpfer in Datenbanken gesammelt. […] Zusätzlich wurden in Zusammenarbeit mit einer internationalen wissenschaftlichen Forschungsgruppe aus Leeds, Cardiff, Peterborough, Nijmengen und Mainz seit 2002 mehr als 3000 gestempelte reliefverzierte Sigillaten aus dem südgallischen La Graufesenque publiziert [MEES, 2011]“, inkl. detaillierter Daten[5] zu Befunden, Töpfermatrizen, Gefäßformen, Produktionsstätten, Formschüsselherstellern und Angaben zur absoluten zeitlichen Datierung. Die südgallische Terra Sigillata ist insbesondere für die Datierung der nördlichen Provinzen von Bedeutung. Die räumliche Verteilung und das zeitliche Muster, sowie die Identifizierung von Produktionszentren und die Rekonstruktion von Handelsnetzen der Töpfer, sind somit zentrale Fragen aktueller archäologischer Forschungen. Der Datensatz ist auf Grund seiner detaillierten Informationen zu Töpfern, Keramik-Fragmenten und deren Fundorten, und anderen Attributen (s.o.) außerordentlich gut geeignet, aktuelle Technologien interoperabler Datenhaltung, insbesondere das

[2] siehe z.B. http://de.wikipedia.org/wiki/Liste_wichtiger_Terra-Sigillata-Gef%C3%A4%C3%9Fformen

[3] vgl. http://de.wikipedia.org/wiki/Terra_Sigillata

[4] vgl. Dragendorfff 1895, 18; Déchelette 1904, 105f., Knorr 1919, VII.

[5] vgl. http://www.rgzm.de/samian

Konzept der Linked Open Data (LOD) und der semantischen Modellierung archäologischer Informationen, exemplarisch zu verdeutlichen. Abbildung 1-1 zeigt einen Ausschnitt dieses Datenmaterials mit der Hervorhebung des Ortes Rheinzabern. Das römische Rheinzabern[6] (Tabernae[7]) war ebenfalls ein bedeutendes Zentrum für Keramikwaren und liegt in der pfälzischen Heimat des Verfassers dieser Arbeit.

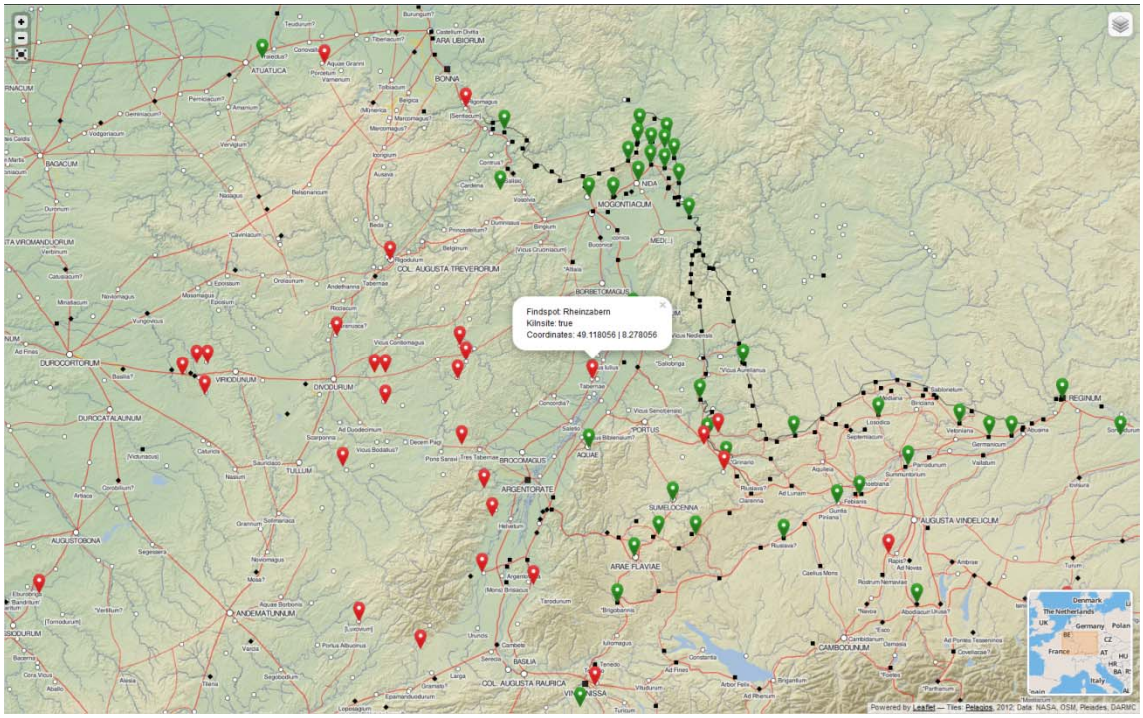


Abbildung 1-1: Terra Sigillata in Zentraleuropa (Datenauszug)

LEIF ISAKSEN zeigt in seiner Dissertation „Archaeology and the Semantic Web“ [ISAKSEN, 2011] potentielle Vor- und Nachteile der Nutzung der Technologie des Web 3.0 in den Geisteswissenschaften (vgl. Abbildung 1-2), hier exemplarisch der Archäologie. Der Einsatz des Semantic Webs ist in den letzten Jahren im Bereich der Kulturwissenschaften signifikant gestiegen. Große Projekte wie das CIDOC CRM[8], Pleiades[9] zeigen den immer stärker werdenden Einfluss von Linked Data in den Geisteswissenschaften. Eine Modellierung der Samian Ware Datenbank als Linked Data könnte somit den Pool an archäologisch relevanten Daten im World Wide Web bereichern.

[6] siehe auch http://www.terra-sigillata-museum.de

[7] siehe auch http://pleiades.stoa.org/places/109362

[8] siehe http://www.cidoc-crm.org

[9] siehe http://pleiades.stoa.org

Weitere Überlegungen zur Archäologie 2.0 und damit zur Kommunikation und Zusammenarbeit verschiedenster archäologischer Repositorien, sind „Archaeology 2.0: New Approaches to Communication and Collaboration[10]“ zu entnehmen.

Criterion	Desirability	Notes
Machine-readable	**	Few archaeologists can program software, but allowing computers to automate simple tasks would clearly be beneficial.
Interactive data	***	A large amount of archaeological research time is spent looking up information about concepts. Instant access to such data would be a significant boon.
Integrated data	***	Archaeology is fundamentally about piecing together fragmentary data.
Inferencing	*	Archaeologists may be sceptical of fully-automated processes of logical deduction.
Data integrity	**	It is accepted that logical inconsistency is inevitable in such a broad discipline. Identifying such inconsistencies may suggest interesting areas of research however.
Self-describing	**	Archaeologists are aware of metadata's importance but not as good at recording it. Automated processes may help.
Persistent global data	*	Few archaeologists are yet willing to make data freely available on the Web, preferring traditional publishing paradigms.
Provenance	***	Knowing the source of archaeological data is very important in judging its utility.
Personalization	*	Archaeologists usually (if not always) intend to be scientifically objective.
Contextualization	***	Providing additional information of potential value is highly helpful to the archaeological process.
Document decomposition	***	The ability to extract and utilize data from grey literature would be extremely useful.
Repurposing	**	Archaeology depends heavily upon prior work but there is a tendency to release 'finished products', rather than raw data, which inhibits repurposing.

Abbildung 1-2: Vorteile der Nutzung des Semantic Web in der Archäologie [ISAKSEN, 2011]

Aufgrund der zuvor erwähnten Vorteile des Semantic Web und der Linked Data in der Archäologie, können diese exemplarisch an dem zuvor beschriebenen Datensatz erprobt werden. Dazu dient folgende Themen- und Aufgabenstellung[11]:

> *„Grundlage der Abschlussarbeit bildet eine Datenbank (Dannell/Mees 2013), in der Töpferstempel auf römischer Keramik, deren Fund- und Herstellungsorte, sowie datierende Merkmale erfasst sind. Neben der hohen Anzahl der gespeicherten Stücke und dem hohen Detaillierungsgrad der Attributinformation zeichnet sich der Datensatz insbesondere durch seine inhärenten Informationen aus, die Aussagekraft zu vielfältigen historischen und archäologische Bezüge besitzen.*

[10] vgl. http://escholarship.org/uc/item/1r6137tb

[11] Vgl. Themenstellung Masterarbeit

Er ist damit außerordentlich gut geeignet, aktuelle Technologien interoperabler Datenhaltung und der semantischen Modellierung archäologischer Informationen exemplarisch vorzuführen. Ziel der Aufgabenstellung ist es neben der Aufbereitung des Datenbestandes und der Implementierung geeigneter Webdienste einerseits, das Potential der Verlinkung heterogener Informationen über Linked Data herauszuarbeiten. Dazu soll die Pelagios-Schnittstelle für die räumliche Verknüpfung anderer Objektgattungen in eine zu entwickelnde, prototypische Webanwendung integriert werden.

Andererseits soll auf Grundlage des Datenbestands eine Ontologie der relativen chronologischen Bezüge der Objekte erarbeitet werden. Dazu sind auf der Basis insbesondere der Ansätze von (Allen 1983) die logischen Relationen in Inferenz- und Integritätsregeln in RDF und OWL zu formulieren und geeignete Visualisierungskonzepte zu evaluieren. Zielsetzung ist eine von absoluten Datierungen unabhängige Beschreibung chronologischer Bezüge der Objekte untereinander.“

- Realisierung der Server-Architektur
- Datenüberführung von Access in PostGIS
    - Formulierung von Exportabfragen aus Access
        - Historisierung (Datum des Exports)
    - Automatisierte Prozessierung von Geometrien aus entsprechenden Attributdaten
- Bereitstellung der Ortsdaten in OGC-konformem Webdienst (WFS, Geoserver)
- Bereitstellung der Töpferstempel, Töpfer, Fragmente, Orte und chronologische Terminologie über eine REST-Schnittstelle
    - Recherche nach und Implementierung von einer geeigneten XML-Auszeichnung
    - Beschreibung der relationalen Logik als flache XML-Hierarchie in den Objekten
- Ergänzende Beschreibung über rdf/owl/ttl
- Entwicklung einer prototypischen Webanwendung
    - Integration der Pleiades API für die Verlinkung der darüber erschlossenen Linked Data Ressourcen
    - Prototypische Entwicklung einer API für Töpfer
- Konzeption und prototypische Umsetzung einer Ontologie zur Repräsentation relativer chronologischer Bezüge archäologischer Objekte (optional)
    - Evaluation von Visualisierungsstrategien zur Darstellung entsprechend modellierter Datenauszüge

2 Technologie und Standards

Dieses Kapitel vermittelt die erforderlichen technischen Grundlagen dieser Arbeit. Datenbanktechnologien, ReSTful Webservices sowie das Semantic Web und Linked Data werden in Grundzügen vorgestellt und hierbei auf weiterführende Literatur verwiesen.

2.1 Standardisierung im Bereich der Geoinformation

Das Open Geospatial Consortium[12] (OGC) ist im Bereich der Geoinformation für eine Reihe von Standards[13] verantwortlich. Es ist ein internationales Industriekonsortium von 473[14] (Stand: 29.11.2013) Firmen, Behörden und Universitäten (z.B. ESRI, Autodesk, Leica Geosystems und das MIT), die durch Teilnahme an Konsensverfahren öffentlich verfügbare Schnittstellenstandards entwickeln. OGG-Standards unterstützen interoperable Lösungen, die das Web im Geo-Bereich bereichern. Die Standards[15] ermöglichen es, Entwicklern komplexe räumliche Informationen und Dienste nutzbar und zugänglich zu machen [OGC, 2013a]. Zu den entwickelten Standards zählen unter anderem GML, CityGML, GeoSPARQL, KML, Simple Features, WMS und WFS [OGC, 2012b]. Weitere Informationen sind der Website www.opengeospatial.org, sowie einer von der OGC bereitgestellten Präsentation[16] zu entnehmen.

Die Geodäsie zeigt ihre Flexibilität und weltweite Anerkennung insbesondere durch verschiedene, auch an lokale Gegebenheiten angepasste, Koordinatenreferenzsysteme, bzw. Projektionen. Diese Daten[17] werden durch die European Petroleum Survey Group Geodesy (EPSG) Arbeitsgruppe europäischer Öl- und Gaserkundungsfirmen in weltweit eindeutigen Schlüsselnummern (EPSG-Codes) abgelegt. Das EPSG wurde 2005 durch das OGP abgelöst [WIKIPEDIA, 2013a]. Bekannte und wichtige EPSG Codes[18] im deutschsprachigen Raum sind 4326[19] (WGS84), 31467[20] (Gauß-Krüger Zone

---

[12] vgl. http://www.opengeospatial.org (abgerufen 03.12.2013)

[13] vgl. http://www.opengeospatial.org/standards/is (abgerufen 03.12.2013)

[14] vgl. http://www.opengeospatial.org/ogc/members (abgerufen 03.12.2013)

[15] vgl. http://www.opengeospatial.org/standards (abgerufen 03.12.2013)

[16] vgl. https://portal.opengeospatial.org/files/?artifact_id=47735 (abgerufen 03.12.2013)

[17] vgl. http://www.epsg.org (abgerufen 03.12.2013) und
http://www.epsg-registry.org (abgerufen 03.12.2013)

[18] vgl. http://spatialreference.org (abgerufen 03.12.2013)

[19] vgl. http://spatialreference.org/ref/epsg/4326 (abgerufen 03.12.2013)

[20] vgl. http://spatialreference.org/ref/epsg/31467 (abgerufen 03.12.2013)

3), 25832[21] (UTM Zone 32), sowie die z. B. in Webkarten genutzte Pseudo-Mercator-Projektion 3857[22]. Das OGC lässt in ihren Standards ebenfalls diese Codes einfließen und ermöglicht so reibungslose Transformationen zwischen mehreren Systemen.

Eine wichtige Standardisierung des OGC bildet die Simple Feature Specification[23] (OGC SFS), die allgemeingültige Regeln und Definitionen für geografische Daten und Geometrien bereitstellt. Sie beschreibt verschiedene geometrische Klassen (vgl. Abbildung 2-1) wie Punkte (Point), Linien (LineString), Polygone (Polygon), mehrere Punkte (MultiPoint), mehrere Linien (MultiLine), mehrere Polygone (MultiPolygon) und eine Sammlung dieser Geometrien (GeometryCollection). Das UML Diagramm verdeutlicht, dass sich alle Klassen von einer abstrakten Geometrie-Klasse ableiten lassen, welche an ein bestimmtes Referenzsystem (vgl. EPSG Codes) gekoppelt ist.

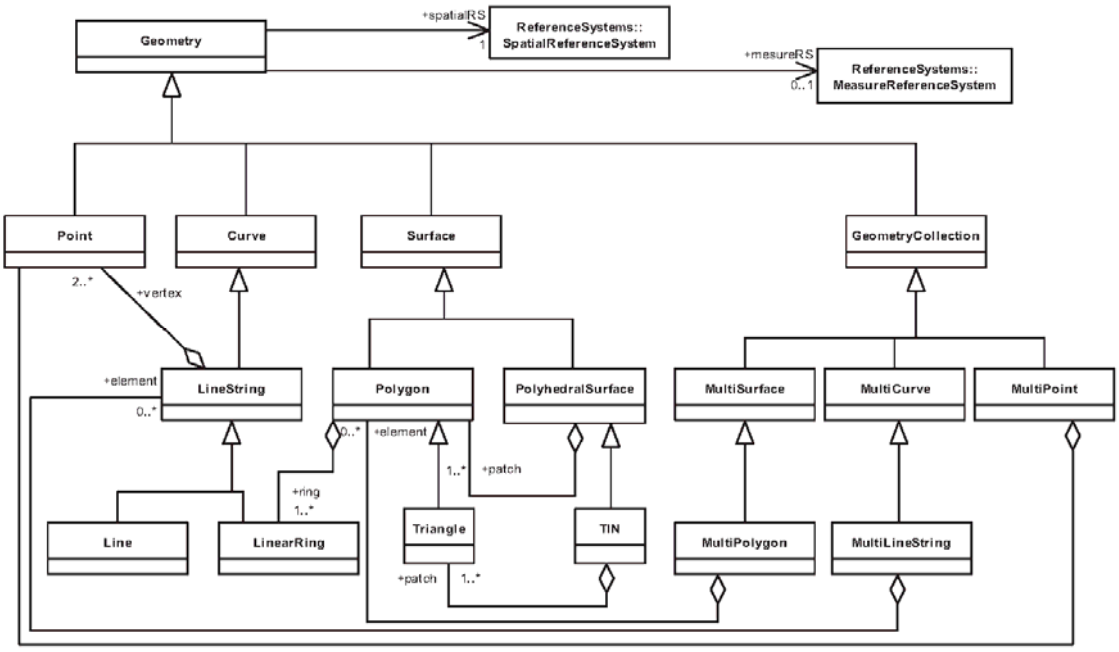


Abbildung 2-1: OGC Simple Feature Specification UML-Diagramm [OGC, 2013c]

Abbildung 2-2 zeigt grafisch die in Abbildung 2-1 gezeigten acht Geometrieklassen. Die OGC SFS kann weiterhin in drei verschiedene Gruppen von Methoden eingeteilt werden: grundlegende Methoden, wie die Abfrage des Geometrietyps, der Ausdehnung oder Geometrie als Well Known Text, Methoden, welche die Beziehung zwischen Objekten beschreiben, sowie Methoden zur räumlichen Analyse wie Buffer und Verschneidungen [OGC, 2013c].

[21] vgl. http://spatialreference.org/ref/epsg/258322 (abgerufen 03.12.2013)

[22] siehe auch http://wiki.openstreetmap.org/wiki/EPSG:3857 (abgerufen 03.12.2013)

[23] vgl. http://www.opengeospatial.org/standards/sfa (abgerufen 03.12.2013)

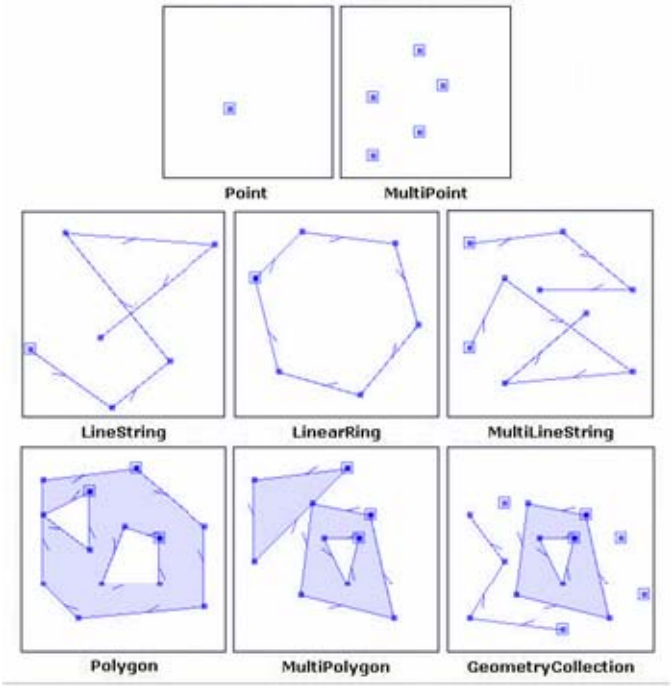


Abbildung 2-2: OGC Simple Features [Key, 2013]

Aus der Simple Feature Specification entstehen Möglichkeiten, wie die Repräsentation von Geometrien in standardisierten XML-Formaten, hier die Geography Markup Language[24] (GML). Weitere Beschreibungen und Erläuterungen zu GML sind Kapitel 3.1 zu entnehmen. Zur Abbildung geografischer und geometrischer Sachverhalte kann ebenfalls das Format JSON genutzt werden. Über die Möglichkeiten der Modellierungen in GeoJSON gibt Kapitel 3.1 weitere Auskünfte.

Der Austausch von Geometrie- inkl. Attributinformationen (Simple Feature) oder Abbildungen von Geometrien kann im Internet durch standardisierte Services des OGC erfolgen, vgl. KORDUAN ET AL. (2008).

Geometrieabbildungen, bzw. Karten können mit Hilfe des Web Map Service[25] (WMS) von einem Server an beliebig viele Clients verschickt werden und somit eine internetgestützte Erstellung von Karten innerhalb eines verteilten Geoinformationssystems (GIS) erfolgen. Ein WMS kann sowohl Raster, wie auch Vektordaten visualisieren. Das Ergebnis wird im Allgemeinen in einem einfachen Bildformat (z.B. PNG) zurückgegeben. Ein OGC-konformer WMS besitzt drei Funktionen, welche vom Benutzer via HTTP angefragt werden können: GetCapabilities, GetMap und getFeatureInfo (optional). Mit GetCapabilities können die Fähigkeiten des WMS erfragt werden. Die Antwort erfolgt in einem XML-Dokument, das vor allem die unterstützten Ausgabeformate und die abfragbaren Layer der Karte beinhaltet. Eine GetMap Anfrage liefert ein georeferenziertes Rasterbild in einem gewünschten Ausgabeformat und ggf. einem bestimmten Ausschnitt. GetFeatureInfo ermöglicht die Abfrage von einem oder mehreren Layern eines WMS [KORDUAN ET AL., 2008].

---

[24] vgl. http://www.opengeospatial.org/standards/gml (abgerufen 03.12.2013)

[25] vgl. http://www.opengeospatial.org/standards/wms (abgerufen 03.12.2013)

Als Erweiterung des WMS kann der Styled Layer Descriptor (SLD) angesehen werden. Diese Erweiterung ermöglicht benutzerdefinierte Systeminformationen an den WMS zu übergeben, der die Karte nach den übergebenen Stilen ausfertigt, vgl. KORDUAN ET AL. (2008).

Der Web Feature Service[26] (WFS) ist im Rahmen der Spezifikationen des OGC das Pendant des Rasterdatendienstes WMS. Dieser liefert Vektordaten über eine HTTP basierende Schnittstelle für den lesenden und schreibenden Zugriff auf geografische Features. Operatoren des WFS sind das Erzeugen, Löschen, Aktualisieren und Löschen von Feature-Instanzen. Ein Feature ist hierbei die Abstraktion eines Faktes der Realität. So kann beispielsweise ein Fluss mit einem Namen, seiner geometrischen Ausdehnung und weiteren Attributen als Feature Type angesehen werden. Für eine Übertragung der Geodaten muss mindestens der Standard GML unterstützt werden, optional können auch weitere Formate implementiert sein (z.B. GeoJSON). Im Vergleich zum WMS unterstützt WFS sechs verschiedene Funktionen: GetCapabilities, DescribeFeatureType, GetFeature, GetGmlObject, Transaction und LockFeature. Eine Sprache, welche Einschränkungen von Abfragen des WFS definiert (Filter), ist die Filter Encoding Implementation Specification des OGC und kann in vier verschiedene Operatoren eingeteilt werden: Vergleichsoperatoren, räumliche Operatoren, logische Operatoren und arithmetische Operatoren. Diese Operatoren können auf Eigenschaften der Features angewendet werden, so ist es z.B. möglich, dass nur Features einer bestimmten Bounding Box oder mit einem bestimmten Attributwert übertragen werden [KORDUAN ET AL., 2008].

Möglichkeiten bei der Erzeugung von Semantik in OGC Diensten, hier insbesondere WFS, können KOHR ET AL. (2012) entnommen werden.

2.2 Datenbanken

In der heutigen Zeit gibt es eine große Anzahl an Datenbanksystemen, die sich beispielsweise im Umfang der Funktionen, der Geschwindigkeit der Verarbeitung, beim Service oder beim Preis unterscheiden. Nachdem vor allem in den neunziger Jahren des vergangenen Jahrhunderts kommerzielle Datenbankhersteller (IBM, Oracle, Microsoft SQL-Server) den Markt beherrschten, gibt es seit der Jahrtausendwende eine größere Anzahl an offenen (freien) Datenbanksystemen. Vor allem Datenbanksysteme wie MySQL (z.B. im Bereich des Webdesign eingesetzt) und PostgreSQL[27] erzielen signifikante Marktanteile. Als Datenbanksystem lässt sich ein computergestütztes System beschreiben, welches mehrere Datenbanken und sämtliche Programme enthält. Eine Datenbank ist eine systematische Sammlung von Daten, welche zentral gespeichert und verwaltet werden sollen. Die zur Verwaltung und Bearbeitung genutzte Software wird als Database-Management-System (DBMS) bezeichnet. Das DBMS stellt unter anderem eine Datendefinitionssprache und eine Datenmanipulationssprache (z.B. SQL[28]) zur Verfügung. Als wichtigste Eigenschaften eines DBMS kann vor allem die Redundanzfreiheit, Datenkonsistenz, Mehrbenutzerfähigkeit und die Datensi-

[26] vgl. http://www.opengeospatial.org/standards/wfs (abgerufen 03.12.2013)

[27] vgl. http://www.postgresql.org (abgerufen 03.12.2013)

[28] siehe auch http://www.torsten-horn.de/techdocs/sql.htm (abgerufen 03.12.2013)

cherung genannt werden. Das wichtigste Datenbankziel ist die Datenunabhängigkeit, welche mit dem Drei-Ebenen-Modell von ANSI/SPARC realisiert ist und folgende Ebenen enthält: Externe Ebene, Konzeptionelle Ebene und Interne Ebene. Heute werden größtenteils nur relationale Datenbanksysteme eingesetzt, da sie für strukturierte Daten und flexiblen Umgang mit Daten besonders geeignet sind. In einem relationalen Datenbanksystem werden die Daten in Form von Tabellen dargestellt und gespeichert. Diese Tabellen werden danach wieder miteinander verbunden, in Relation gebracht. Die Definition dieser Datenstruktur, sowie die Abfrage, können mittels der relationalen Abfragesprache SQL (Structured Query Language) erfolgen. Einige Datenbankanbieter verfügen über objektorientierte Eigenschaften und werden als objektrationale Datenbanken bezeichnet. PostgreSQL ist eines der weitverbreitetsten objektrationalen Datenbanksysteme im Open Source Bereich. PostgreSQL unterstützt Programmierkonstrukte, wie gespeicherte Prozeduren, Trigger und Cursor und verfügt über Schnittstellen zu Programmiersprachen wie JJAVA und C/C++. Ein Vergleich mit kommerziellen Datenbanken in Puncto Leistung und Flexibilität ist statthaft. PostgreSQL ermöglicht den Benutzern eine Erweiterung des Systems um eigene Datentypen, Operatoren und Funktionen. Einen Austausch des Server-DBMS gegenüber einer unabhängigen Client-Applikation setzt ein Client/Server Konzept voraus. PostgreSQL realisiert ein funktional verteiltes System, in dem die Kommunikation über eine Schnittstelle realisiert wird. Die Schnittstelle zwischen Client und Server ist in diesem Fall das Datenbankverwaltungssystem, vgl. auch Kapitel 4 [PAPAKOSTAS, 2010].

Eine Datenbank kann als ein Abbild eines Teils der realen Welt betrachtet werden (z.B. Kunden und deren Kundenadressen). Eine der wichtigsten Anforderungen an einen Datenbankentwurf ist es, dass nicht alle Daten in einer einzigen Tabelle abgelegt werden. Eine Tabelle wird im relationalen Datenbankmodell durch einen eindeutigen Namen gekennzeichnet und besteht aus Zeilen und Spalten (Attribute). Beim Entwurf einer Datenbank ist folgendes zu beachten: Jeder Bereich einer Datensammlung muss mit einer Tabelle abgebildet werden, um eine Datenredundanz (mehrfaches Vorkommen von gleichen Daten) zu vermeiden. Jedes Feld einer Tabelle sollte durch einen charakteristischen Datentyp gekennzeichnet werden und nur eine Art von Daten enthalten, sodass gezielt nach ihnen gesucht werden kann. Ein Primärschlüssel, der jeder Tabelle zugeordnet ist, lässt jeden Datensatz einer Tabelle eindeutig identifizieren. Eine Tabelle zeichnet sich zudem dadurch aus, dass sie keine bis beliebig viele Zeilen enthalten kann, welche allerdings immer unterschiedlich sein müssen, so dass keine Tupel (zwei gleiche Zeilen) entstehen [PAPAKOSTAS, 2010].

Die Datenredundanz ist ein großes Problem bei einem Entwurf einer Datenbank. Besonders die unnötige Speicherplatzbelegung und die komplexe und unübersichtliche Aktualisierung erzeugen Nachteile. Auch die sogenannte Löschanomalie (z.B. Löschung eines Kunden und dadurch Löschung der dazugehörigen Bestellung) und Änderungsanomalie (Mehrfachänderung von Sachverhalten) sind negativ zu bewerten. Eine Normalisierung von Tabellen ermöglicht die Vermeidung einer Datenredundanz. Übergreifende Informationseinheiten werden hiermit in kleinere Einheiten zerlegt, bis eine doppelte Speicherung nahezu unmöglich ist. Ein Nebeneffekt hierbei ist eine übersichtliche Darstellung der Tabellen. Der Prozess der Normalisierung[29] umfasst mehrere Normal-

[29] siehe auch http://www.tinohempel.de/info/info/datenbank/normalisierung.htm (abgerufen 03.12.2013)

formen, auf welche an dieser Stelle nicht eingegangen wird. Nach erfolgreicher Normalisierung können Beziehungen zwischen Tabellen erzeugt werden: 1:1 Beziehung, 1:N Beziehung und N:M Beziehung, vgl. auch Vorgehen in Kapitel 4 [PAPAKOSTAS, 2010].

PostgreSQL bietet die Möglichkeit durch Standardfunktionen wie mathematische Funktionen, Zeichenkettenfunktionen und Datums- bzw. Zeitfunktionen, Abfragen via SQL zu tätigen. Des Weiteren können eigene Funktionen und Trigger hinterlegt werden. Bestimmte Sichten auf Datenbestände (sog. Views) beschreiben logische Datenschichten zu real hinterlegten Tabellendaten. Daher werden diese auch als virtuelle Tabellen bezeichnet, da keine physischen Sichtdaten existieren [PAPAKOSTAS, 2010].

In dieser Arbeit werden sowohl die relationalen Datenbanksysteme Microsoft Access (auf Seiten des RGZM), als auch PostgreSQL mit einem PostGIS-Aufsatz genutzt.

Die in dieser Masterarbeit verwendeten Daten (vgl. Kapitel 4) zeichnen sich durch eine geografische Komponente (einen Koordinatenpunkt eines Findspot) aus. Räumliche Abfragen sind in einer konventionellen relationalen Datenbank wie PostgreSQL nicht möglich. In diesem Fall muss über den Einsatz einer Spatial Database nachgedacht werden. Als Spatial Database kann eine Datenbank betrachtet werden, die spezielle Typen zur Speicherung geometrischer Objekte in einer normalen Datenbank bereitstellt. Sie bietet zudem spezielle Funktionen an, die das Abfragen und Manipulieren z.B. mit SQL ermöglicht [OBE ET AL., 2011].

In dieser Arbeit wird eine PostGIS 2.0[30] Datenbank genutzt, die eine Erweiterung der objektrationalen PostgreSQL Datenbank darstellt. PostGIS wird von der Refractions Research Inc. entwickelt, einem Forschungsprojekt im Umfeld von räumlichen Datenbanktechnologien. Refractions ist ein GIS-und Datenbankberatungsunternehmen mit Sitz in Victoria, British Columbia, Kanada, spezialisiert auf Datenintegration und Entwicklung kundenspezifischer Software [POSTGIS, 2013].

Mit Hilfe von PostGIS 2.0 können verschiedene Geometrietypen gespeichert und abgefragt[31] werden werden: point, line, polygon, multipoint, multiline, multipolygon, and geometrycollections, sowie [POSTGIS, 2013]:

- den „standard OGC geometry data type“, der ein ebenes Koordinatensystem für Messungen benutzt
- den „geography data type”, der ein geodätisches Koordinatensystem benutzt.
- den “raster”-Typ.

Im Rahmen dieser Arbeit wird der OGC-Typ verwendet. Die OpenGIS Spezifikation erlaubt zudem die Speicherung der Geometrie in verschiedenen ‚spatial referencing systems‘ (SRID: Spatial Reference IDentifier), vgl. POSTGIS (2013).

[30] vgl. http://postgis.net (abgerufen 03.12.2013)

[31] vgl. http://postgis.net/docs/manual-2.0/reference.html (abgerufen 03.12.2013)

PostGIS mit PostgreSQL ist nicht die einzig existierende Spatial Database. Die meisten kommerziellen relationalen High-End-Datenbanken bieten Geo-Funktionalitäten (z.B. Oracle und IBM). Insbesondere SpatiaLite, welches ein Add-On zur portablen Open Source SQLite[32] Datenbank bildet (und z.B. auch in QGIS verfügbar ist), bietet eine Low-End Alternative [OBE ET AL., 2011]. In dieser Arbeit wird jedoch PostGIS aufgrund seiner Client-Server-Struktur und der umfangreichen Möglichkeiten der Nutzung verwendet.

2.3 Geoserver

Geoserver[33] ist eine Open Source Software (Webserver), geschrieben in JAVA, die es Nutzern erlaubt Geodaten zu editieren und zu veröffentlichen. An der Entwicklung des Geoservers ist eine große Community, verteilt über die ganze Welt, beteiligt. Interoperabilität ist eines der Hauptprinzipien des Geoservers, so dass die Nutzung von offenen Standards (vgl. OGC) Grundvoraussetzung ist. Des weiteren ist Geoserver eine Referenzimplementierung der vom Open Geospatial Consortium (OCG) implementierten Standards Web Feature Service (WFS), Web Coverage Service (WCS) und Web Map Service (WMS). Somit bildet der Geoserver eines der Hauptkomponenten des Geospatial Web [GEOSERVER, 2013A]. Das Open Source GIS-Toolkit GeoTools[34] bildet das Herzstück der Anwendung [GEOSERVER, 2013B], das dadurch eine Vielzahl von weiteren Features[35] bereitstellt. Nähere Informationen zur Implementierung (der verwendeten Datenbasis etc.) sind Kapitel 6.2 zu entnehmen.

2.4 ReSTful Webservices

ReSTful Webservices beinhalten das Akronym ReST (oder auch REST beschrieben), das für Representional State Transfer steht [FIELDING, 2000]. ReST ist ein Architekturstil, welcher ganze Datenbestände oder einzelne Ressourcen sowohl für Menschen, als auch für Maschinen in bearbeitbarer Form zur Verfügung stellt und auf eine Dissertation von Roy Fielding zurückzuführen ist. Im Vergleich zu den in Kapitel 2.1 beschriebenen OGC Spezifikationen, ist ReST kein Standard, eher eine Richtlinie, vgl. HAFT (2013). Zahlreiche Unternehmen wie Twitter, Facebook und Google bieten mittlerweile ReST-Schnittstellen zu ihren Diensten an [HELMICH, 2013].

Das World Wide Web besteht aus Sicht menschlicher Benutzer nur aus Komponenten, die er selbst wahrnehmen kann, z.B. Webapplikationen, die über einen Browser aufgerufen werden können. Über diese HTTP Schnittstelle greifen jedoch nicht nur das menschliche Auge, sondern auch andere Anwendungen zu. Stellt ein Betreiber einen bestimmten Dienst anderen Benutzern zur Verfügung spricht man von einem Webservice. Zur Zeit der Jahrtausendwende wurde für solche Webser-

---

[32] vgl. http://www.sqlite.org (abgerufen 03.12.2013)

[33] vgl. http://geoserver.org(abgerufen 03.12.2013) und http://live.osgeo.org/de/overview/geoserver_overview.html (abgerufen 03.12.2013)

[34] vgl. http://www.geotools.org (abgerufen 03.12.2013)

[35] vgl. http://geoserver.org/display/GEOS/Features (abgerufen 03.12.2013)

vices das Simple Object Access Protokoll[36] (SOAP) verwendet. SOAP ermöglicht den Aufruf von Methoden eines serverseitigen Objekts mit Hilfe von komplexen XML-Nachrichten via HTTP-Austausch. Die durch eine Kombination von XML und HTTP erreichte Programmiersprachen- und Plattformunabhängigkeit zeigt jedoch einige Schwächen, wie einen recht großen Overhead (viele Metadaten, wenig Nutzdaten) und das rechenintensive Zusammen- und Auseinanderbauen der XML-Nachrichten. Eine Erweiterung des SOAP in viele kleinere WS-* Standards[37] erschwert den Zugriff und Überblick der Möglichkeiten erheblich [HELMICH, 2013].

Eine Vereinfachung des komplexen SOAP Standards stellt der ReST-Architekturstil dar, vgl. FIELDING, 2000. Nach Fielding ist ReST unabhängig von konkreten Protokollen, zumeist wird jedoch heutzutage das anerkannte HTTP-Protokoll[38] verwendet. Der fundamentale Bestandteil der ReST-Architektur ist die Ressource [HELMICH, 2013]. Eine Ressource ist ein im Internet ansprechbares Objekt eines bestimmten Typs und damit assoziierten Daten mit Beziehungen zu anderen Ressourcen. Sie ist ähnlich einer Objektinstanz in einer objektorientierten Programmiersprache mit dem großen Unterschied, dass nur ein paar bestimmte Standardmethoden zur Verfügung gestellt werden [JANSEN, 2012]. Twitter[39] beispielsweise nutzt ReSTful Webservices. So könnte jeder Benutzer des sozialen Netzwerks und jeder einzelne Tweet als eine Ressource betrachtet werden. In dieser Arbeit wird jeder einzelne Töpfer, Findspot und jedes Fragment als Ressource gesehen und ist über einen eindeutigen URI[40] ansprechbar. Laut Fielding sollen Ressourcen folgende Anforderungen erfüllen [HELMICH, 2013].

1. Adressierbarkeit: Eine Ressource muss über einen eindeutigen Unique Ressource Identifier (URI) identifiziert werden können.
2. Zustandslosigkeit: Eine Kommunikation der Teilnehmer ist zustandslos, das heißt, dass keine Benutzersitzungen (z.B. Sessions oder Cookies) existieren und bei jeder neuen Anfrage alle notwendigen Informationen erneut gesendet werden müssen.
3. Einheitliche Schnittstelle: Jede Ressource muss mit einem einheitlichen Satz von Standardmethoden zugreifbar sein, z.B. GET, POST, PUT, etc.
4. Entkopplung von Ressourcen und Repräsentation: Eine Ressource kann in mehreren Repräsentationsformen vorliegen, z.B. HTML, XML und JSON.

Ressourcen können gruppiert werden, in so genannte Collections. Jede Collection ist homogen, das heißt, dass sie nur einen Typ einer Ressource und untergeordnete Ressourcen enthält, vgl. Abbildung 2-3 [Jansen, 2012].

---

[36] vgl. http://www.w3.org/TR/soap (abgerufen 03.12.2013)

[37] siehe auch http://de.wikipedia.org/wiki/WS-* (abgerufen 03.12.2013)

[38] vgl. http://www.w3.org/Protocols (abgerufen 03.12.2013)

[39] vgl. https://twitter.com (abgerufen 03.12.2013)

[40] siehe auch http://de.wikipedia.org/wiki/Uniform_Resource_Identifier (abgerufen 03.12.2013)

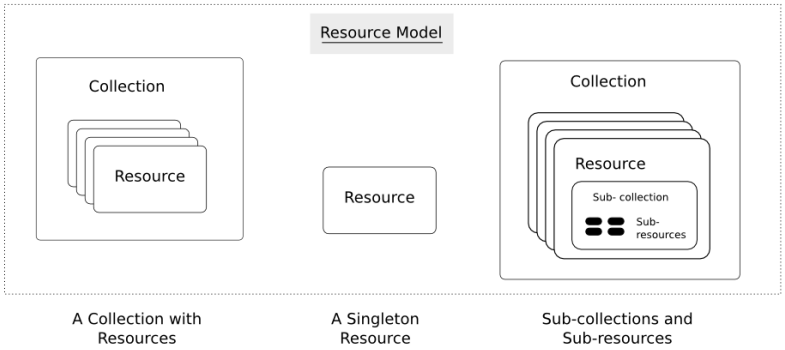


Abbildung 2-3: Ressourcen und Collections [Restful, 2013]

Ressourcen sollten bestimmte Metadaten enthalten: ID, href und link. Als ID ist ein eindeutiger Bezeichner einer Ressource anzusehen. Href identifiziert die URl in der gegenwärtigen Ressource, link gibt eine Beziehung einer Ressource an, siehe Abbildung 2-4. Darüber hinaus werden die nachfolgenden Repräsentationsformen vorgeschlagen: HTML, XML, JSON, YAML [Jansen, 2012].

```
<collection id="samian" href="http://143.93.114.104/rest/samian">
  <resources id="potters" href="http://143.93.114.104/rest/samian/potters"/>
  <resources id="findspots" href="http://143.93.114.104/rest/samian/findspots"/>
  <resources id="fragments" href="http://143.93.114.104/rest/samian/fragments"/>
</collection>
```

Abbildung 2-4: ReST-Beispiel[41]

Der HTTP-Standard definiert einen ganzen Satz an Operationen, mit denen auf Ressourcen zugegriffen werden kann. Die Methoden GET und POST werden auch von jedem Browser verwendet, wenn Internetseiten aufgerufen, bzw. Formulare abgeschickt werden [HELMICH, 2013]. Daneben gibt es aber auch noch einen ganzen Satz weiterer Operationen, die von den meisten Browsern nicht genutzt werden: PUT, DELETE, PATCH sowie HEAD und OPTIONS, vgl. JANSEN (2012) und RICHARDSON ET AL. (2007).

GET dient zum lesenden Zugriff auf Ressourcen ohne die Daten auf dem Server zu verändern. Mit einem POST-Request können neue Daten erstellt, mit einem PUT Request Daten erstellt oder bearbeitet und mit DELETE gelöscht werden [HELMICH, 2013]. Die Methoden HEAD und OPTIONS ermitteln Metadaten bzw. übermitteln mögliche HTTP Methoden, die verfügbar sind [RICHARDSON ET AL., 2007]. In der Regel sollten all diese Methoden in einen ReSTful Webservice integriert werden [JANSEN, 2012].

Es zeigt sich, dass mit einer ReST-Schnittstelle sowohl ein Datenbestand befragt, geändert und gelöscht werden kann, ohne dass die Struktur der Daten bekannt sein muss. Es ist allerdings anzumerken, dass u. A. die Methoden POST und PUT unscharf definiert sind. Ein PUT einer noch nicht existierenden Methode kann eine Erstellung oder eine Änderung einer Ressource zur Folge haben.

[41] Quelle: http://143.93.114.104/rest/samian (abgerufen 03.12.2013)

Gleiche Eigenschaften können der Methode POST zugesprochen werden. Hier zeigen sich Probleme der fehlenden Standardisierung. Eine Rückgabe eines bestimmten Datenformats kann mit Hilfe einer Information im Header einer http-Anfrage (z.B. Accept: application/xml) gesteuert werden [HAFT, 2013].

Eine ReSTful API benötigt genau einen Entry Point. Die URL des Entry Points muss auf den Client übermittelt werden, so dass die API gefunden werden kann. Der Entry Point sollte folgende Informationen enthalten: Informationen über die API Version und unterstützte Features, eine Liste von Collections und eine Liste an Ressourcen, sowie weitere Informationen, die als nützlich erachtet werden. Jede Collection und jede Ressource besitzt ihre eigene URL. URLs sollten bestenfalls nie von einem Client aufgebaut werden, sondern nur Links (link, href) der API folgen. Eine empfohlene Konvention in Bezug auf URLs ist es, alternierende Pfade von Ressourcen- und Collection-Elementen zu verwenden, vgl. Tabelle 2-1 [JANSEN, 2012]:

Tabelle 2-1: Aufbau einer ReST-URL[42]

URL	Description
/api	The API entry point
/api/:coll	A top-level collection named “coll”
/api/:coll/:id	The resource “id” inside collection “coll”
/api/:coll/:id/:subcoll	Sub-collection “subcoll” under resource “id”
/api/:coll/:id/:subcoll/:subid	The resource “subid” inside “subcoll”

Der Entry Point, der in dieser Arbeit erzeugten ReST-Schnittstelle (vgl. Kapitel 6.1) ist http://143.93.114.104/rest. Über diesen Einstiegspunkt sind alternierende Pfade zu Collections und Ressourcen erreichbar. Die implementierte HTTP GET Methode erzeugt verschiedene Ressourcen- und Collection-Repräsentationen wie HTML, XML, JSON, aber auch RDF Dialekte (vgl. Kapitel 6.1). Die ReST-Schnittstelle dieser Arbeit stellt eine vorläufige, prototypische Version der ReST-Schnittstelle dar. Im Fall der Veröffentlichung muss gegebenenfalls erneut über das Design der ReST-ful URIs[43] nachgedacht werden. Ein gutes Design der URIs kann andere Websites von der Nutzung der ReST-Schnittstelle überzeugen und die Benutzerfreundlichkeit erhöhen [2PARTSMAGIC, 2013].

[42] Quelle: http://restful-api-design.readthedocs.org/en/latest/urls.html (abgerufen 03.12.2013)

[43] siehe auch http://blog.2partsmagic.com/restful-uri-design (abgerufen 03.12.2013)

Ein Beispiel einer bereits implementierten und veröffentlichten ReST-Schnittstelle nach den vorgestellten Richtlinien im Bereich der Digital Humanities ist das ReST-Interface[44] der Deutschen Inschriften Online[45]. Hier ist die Struktur der Collections[46] und Ressourcen[47], inkl. der weiterführenden Links und IDs umgesetzt und ermöglicht ein computergestütztes Auslesen des Datenbestands. Dies wird beispielsweise im Rahmen des Projekts 'Inschriften im Bezugssystem des Raums[48] (IBR)' durch JavaScript und jQuery gelöst[49].

2.5 Semantic Web und Linked Data

Die Worte Semantic Web und Linked Data, bzw. Web 3.0, sind in den Digital Humanities[50] in voller Munde. Die nachfolgenden Kapitel geben einen Einblick in dessen Möglichkeiten und Technologien. Insbesondere wird hierbei auf das Ressource Description Framework (RDF), dessen Abfragesprache (SPARQL), Speichermedien für Linked Data (Triple Stores) und die Ontologiesprache OWL eingegangen.

2.5.1 Semantic Web

Die Idee des Semantic Web[51] beruht unter anderem auf Arbeiten von Tim Berners-Lee [BERNERS-LEE ET AL., 2001]. Berners-Lee[52] selbst ist Direktor des World Wide Web Consortium[53] (W3C). Hauptziel des W3C ist die Entwicklung, Ausbildung und Verarbeitung von Web-Standards[54] und Richtlinien (z.B: HTML[55], XML[56] und PNG[57]), so genannte Web-Interoperabilität. Es wurde 1994 vom Erfinder des World Wide Web[58] Berners-Lee als internationales Konsortium verschiedener

---

[44] vgl. http://www.inschriften.net/rest/ (abgerufen 03.12.2013)

[45] vgl. http://www.inschriften.net (abgerufen 03.12.2013)

[46] z.B. http://www.inschriften.net/rest/di019 (abgerufen 03.12.2013)

[47] z.B. http://www.inschriften.net/rest/di019/articles/di019-0001 (abgerufen 03.12.2013)

[48] vgl. http://www.spatialhumanities.de/ibr/startseite.html (abgerufen 03.12.2013)

[49] siehe auch http://v.spatialhumanities.de. Dies ist die Testseite des Generic Viewers, welcher u. A. Fachdaten (Inschriften) der Deutschen Inschriften Online (http://www.inschriften.net/rest) und räumliche Informationen (http://s.spatialhumanities.de) aus ReST-Schnittstellen auslesen und dem Anwender zur Verfügung stellen kann (Stand: 29.11.2013).

[50] vgl. http://www.dig-hum.de (abgerufen 03.12.2013)

[51] siehe auch http://semanticweb.org/wiki/Main_Page (abgerufen 03.12.2013)

[52] siehe auch http://www.w3.org/People/Berners-Lee (abgerufen 03.12.2013)

[53] vgl. http://www.w3.org (abgerufen 03.12.2013)

[54] vgl. http://www.w3.org/TR (abgerufen 03.12.2013)

[55] vgl. http://www.w3.org/TR/html-markup (abgerufen 03.12.2013)

[56] vgl. http://www.w3.org/TR/xml (abgerufen 03.12.2013)

[57] vgl. http://www.w3.org/TR/PNG (abgerufen 03.12.2013)

[58] hier der Originalvorschlag: http://www.w3.org/History/1989/proposal.html (abgerufen 03.12.2013)

Arbeitgeber und Organisationen[59] gegründet. [W3C, 2013]. Die Technologie des Semantic Web[60] wird auch als Web 3.0 bezeichnet [ITWISSEN, 2013].

Das Web 1.0[61] hatte lediglich die Möglichkeit Informationen und Daten auf statischen Seiten bereitzustellen und nur einen lesenden Zugriff auf die Daten zu geben. Eine Interaktion mit dem Benutzer wird auf ein Minimum reduziert, da nur Informationen abgerufen und gesucht werden können, die auf den Webseiten verfügbar sind. Informationen wurden zumeist von Intuitionen und Firmen zur Verfügung gestellt, so dass hier oft eine hohe inhaltliche Qualität erreicht werden kann. Im Vergleich zum Web 1.0 erlaubt das Web 2.0[62] eine Interaktion mit Websites und Nutzern, sowie auch anderen Websites, bzw. nicht nur einen lesenden, sondern auch einen schreibenden Zugriff. Als Repräsentanten dieses Webs können Youtube[63], Facebook[64] und MySpace[65] oder Wikipedia[66] genannt werden. Insbesondere der Aufstieg von sozialen Netzwerken (Twitter[67], Facebook, bzw. Google+[68]) kann dem Web 2.0 angerechnet werden (vgl.

Abbildung 2-5). Nutzer können hierbei Inhalte selbst erstellen, Informationen mit anderen (einer Community) teilen und an Informationen anderer teilhaben. Als Negativ muss hierbei angemerkt werden, dass Inhalte (oft) nicht kontrolliert werden können und die inhaltlich hohe Qualität durch die Interaktivität nur schwer erreicht werden kann. Das Web 3.0[69] kann als neues Zeitalter in der Internettechnologie angesehen werden, das durch einfache Schnittstellen eine umfassende Suche nach Informationen ermöglicht. Es bietet hochgenaue Suchen nach komplexen benutzerspezifischen Kriterien an. Das Web 3.0 schafft eine Sammlung von Repositorien, welche mit den eigenen Daten verknüpft werden können. Jedoch ist auch hier die Gefahr gegeben, dass nicht jede Anfrage ein genaues Ergebnis liefert [AYONAKAY, 2012A].

---

[59] siehe http://www.w3.org/Consortium/Member/Testimonial/List (abgerufen 03.12.2013)

[60] siehe auch (abgerufen 03.12.2013)
http://www.spiegel.de/netzwelt/web/semantic-web-das-internet-soll-klueger-werden-a-561831.html

[61] siehe auch http://www.itwissen.info/definition/lexikon/Web-web.html (abgerufen 03.12.2013)

[62] siehe auch http://www.itwissen.info/definition/lexikon/Web-2-0-web-2-0.html (abgerufen 03.12.2013)

[63] vgl. http://www.youtube.com (abgerufen 03.12.2013)

[64] vgl. https://www.facebook.com (abgerufen 03.12.2013)

[65] vgl. https://myspace.com (abgerufen 03.12.2013)

[66] vgl. http://de.wikipedia.org (abgerufen 03.12.2013)

[67] vgl. https://twitter.com (abgerufen 03.12.2013)

[68] vgl. https://plus.google.com (abgerufen 03.12.2013)

[69] siehe auch http://www.itwissen.info/definition/lexikon/Web-3-0-web-3-0.html (abgerufen 03.12.2013)

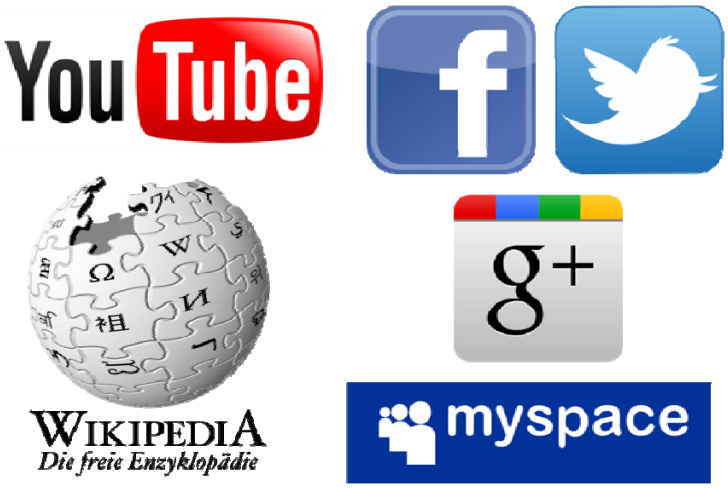


Abbildung 2-5: Anwendungen des Web 2.0

Abbildung 2-5 zeigt einen Vergleich der drei vorgestellten Versionen des Word Wide Web. Die in den vorherigen Versionen wichtigen dynamischen Nutzerinteraktionen (siehe soziale Netzwerke) stehen im Web 3.0 im Hintergrund. Mehr stehen semantische, das heißt bedeutungstragende Verknüpfungen von Informationen und Inhalten untereinander, im Fokus. Hierdurch entsteht auch die Möglichkeit, automatisch neue Informationen abzuleiten bzw. herzustellen. Dazu muss der Computer in die Lage versetzt werden, automatisch Informationen zu erfassen, verstehen zu verarbeiten. „Das Semantic Web steht [somit] für die Idee die Informationen von vornherein in einer Art und Weise zur Verfügung zu stellen, die deren Verarbeitung durch Maschinen ermöglicht [HITZLER ET AL., 2008]“. Die Technik und Sprache der Verarbeitung ist Linked Data (vgl. Kapitel 2.5.2) und RDF (vgl. Kapitel 2.5.3).

Crawl	Walk	Run
Web 1.0	Web 2.0	Web 3.0
Mostly Read-Only	Wildly Read-Write	Portable & Personal
Company Focus	Community Focus	Individual Focus
Home Pages	Blogs / Wikis	Lifestreams / Waves
Owning Content	Sharing Content	Consolidating Content
Web Forms	Web Applications	Smart Applications
Directories	Tagging	User Behavior
Page Views	Cost Per Click	User Engagement
Banner Advertising	Interactive Advertising	Behavioral Advertising
Britannica Online	Wikipedia	The Semantic Web
HTML / Portals	XML / RSS	RDF / RDFS / OWL

Abbildung 2-6: Vergleich der Versionen des Web [AYONAKAY, 2012B]

Der Übergang von der Industrie- zur Informationsgesellschaft findet die markanteste Ausprägung in der Entwicklung im World Wide Web. Die Chance jeder Zeit auf dem neusten Stand der Information, mit einer vernachlässigbaren Übertragungszeit zum Nutzer zu sein, bietet die Möglichkeit der ständigen Aktualität und universelle Verfügbarkeit von Informationen. Das Web hat weiterhin zu einer Liberalisierung der bereitgestellten Informationen geführt (z.B. gedrucktes Buch im Vergleich zu Online-Publikationen). Die Erschaffung virtueller Marktplätze und Auktionsplattformen (z.B. Amazon und eBay) sind ein weiteres Phänomen unserer Informationsgesellschaft. Diese Entwicklung birgt jedoch auch Probleme in sich. Besonders die unüberschaubare Menge an präsenten Informationen und Repräsentationsformen, die rein auf den Menschen als Benutzer ausgelegt sind, stellen ein Problem dar. Viele Informationen werden zudem zur Verfügung gestellt, jedoch von keiner Suchmaschine (z.B. Google, Yahoo, etc.) erfasst. Zurzeit ist noch eine stichwortartige Suche in Dokumenten des Webs realisiert. Wünschenswert wären jedoch inhaltliche, das heißt semantische Suchabfragen. Auch die Heterogenität des Webs (dezentrale Struktur und Organisation des Webs) zeigt Probleme auf. Unterschiedliche Datenformate bzw. Kodierungstechniken und verschiedene natürliche Sprachen machen es fast unmöglich über das Web verteilte Informationen zu sammeln. Eine einheitliche Aufbereitung und Weiterverwendung ist kaum möglich und zeigt somit das Problem der Informationsintegration. Die Problematik der Extraktion impliziten Wissens zeigt weitere Probleme auf. Die Lösung der Probleme könnte das Semantic Web sein [HITZLER ET AL., 2008].

Man kann zwei komplementäre Verfahren zur Lösung der Probleme unterscheiden: Zum einen kann man Methoden der künstlichen Intelligenz nutzen, um damit primär für Menschen repräsentierte Informationen in formale, das heißt maschinenlesbare Daten, umzuwandeln. Die bisher erreichten Resultate zeigen jedoch, dass in näherer Zukunft eine Lösung nicht in Sicht ist. Zum anderen kann der Ansatz des Semantic Web weiterverfolgt werden, sodass Informationen von vornherein in Formaten zur Verfügung stehen, die maschinenlesbar sind. Hierbei ist es nötig, offene Standards zu definieren und zu verwenden (vgl. XML, RDF, RDFs und OWL des W3C), sodass Informationen zwischen mehreren Plattformen ausgetauscht und in Beziehung mit einander gesetzt werden können (= Interoperabilität). Methoden, die zur Schlussfolgerung von neunen Informationen dienen (formale Logik) sind ein weiterer wichtiger Bestandteil des Semantic Web [HITZLER ET AL., 2008]. Einfach gesagt können die Zeile des Semantic Web wie folgt zusammengefasst werden: „Finde Wege und Methoden, Informationen zu repräsentieren, dass Maschinen damit in einer Art und Weise umgehen können, die aus menschlicher Sicht nützlich und sinnvoll erscheint. [HITZLER ET AL., 2008]".

In der bereits zuvor erwähnten Informationsgesellschaft kann das Dokument als zentrale Einheit gesehen werden. Jede Seite des klassischen World Wide Web (1.0 und 2.0) ist durch Hyperlinks mit anderen Dokumenten verbunden. Suchmaschinen stellen mit Hilfe von Information-Retrieval und beispielsweise dem Google PageRank Algorithmus[70] Ranglisten auf, die zum einen auf die Anzahl an Links zur Website und zum anderen auf den Page Rank der verlinkten Seiten basiert. Das Web 1.0 kann somit auch als ein Web of Documents beschrieben werden. Datendienste bzw. Schnittstel-

[70] siehe auch http://www-i1.informatik.rwth-aachen.de/~algorithmus/algo10.php (abgerufen 03.12.2013)

len wie ReST (vgl. Kapitel 2.4) und Standardisierungen wie XML oder RDF und die Veröffentlichung von Informationen in diesen strukturierten Formaten, ist ein Schritt hin zu einem Web of Knowledge. Das Semantic Web kann somit als ein Web von Informationen angesehen werden, da die Semantik von Informationen publiziert wird und ein Ableiten neuer Inhalte möglich ist [HART ET. AL, 2013].

Eine weitere (informelle) Definition des Semantic Web kann BERNERS-LEE ET AL. (2001) entnommen werden: „The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.". Daraus lässt sich der von Tim Berners-Lee entwickelte Semantic Web Tower, bzw. Semantic Web Layercake[71] ermitteln (vgl. Abbildung 2-7). Es zeigt eine Illustration der Hierarchie verschiedener Sprachen, wobei jede Schicht Funktionen der unteren Schichten nutzt. Des Weiteren ist zu sehen, wie Technologien standardisiert sind, so dass die Entwicklung des Semantic Web möglich ist und verweist auf die Verbindung zum klassischen Hypertext Web. Die oberen Schichten (Logic, Proof und Trust) sind jedoch nicht standardisiert und zeigen lediglich Ideen auf, wie das Semantic Web realisiert werden kann. Insbesondere die Verschlüsselung und Überprüfung der Daten stellt noch ein Problem dar.

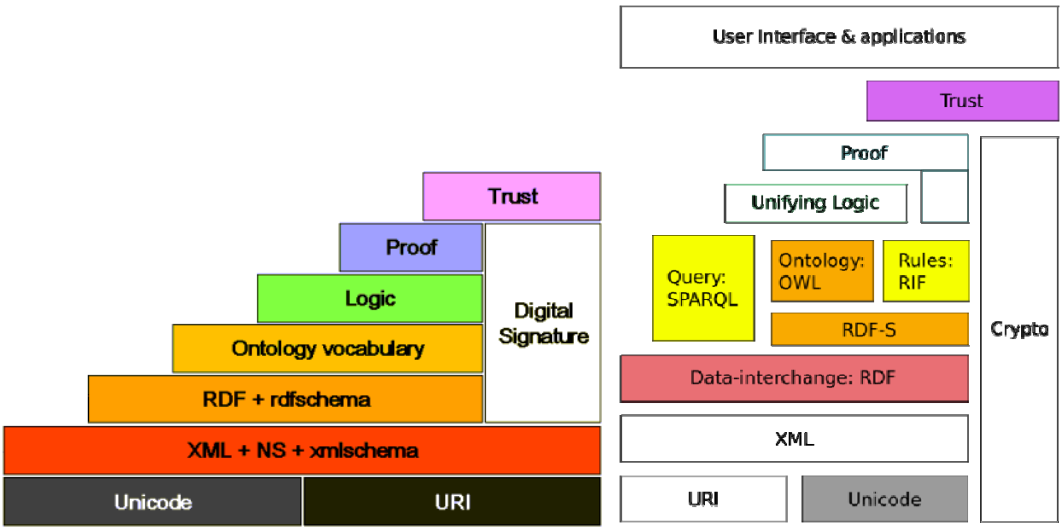


Abbildung 2-7: Semantic Web Tower[72]

2.5.2 Linked (Open) Data

Wie bereits in Kapitel 2.5.1 erläutert, stellt das Semantic Web nicht allein ein Web von Daten dar. Eine Verlinkung dieser, so dass Personen oder Maschinen das entstehende Web of Data erkunden können, ist der zentrale Aspekt der Linked Data[73]. Der Begriff Linked Data[74] geht auf Tim Berners-

[71] siehe auch http://en.wikipedia.org/wiki/Semantic_Web_Stack (abgerufen 03.12.2013)

[72] Quellen: http://www.w3.org/RDF/Metalog/docs/sw-easy (abgerufen 03.12.2013) und http://commons.wikimedia.org/wiki/File:SW_layercake_2006.svg#file (abgerufen 03.12.2013)

[73] siehe auch http://www.w3.org/standards/semanticweb/data (abgerufen 03.12.2013)

Lee zurück[75]. Entstehen viele Repositorien mit Linked Data, so sind eine Suche und das Auffinden anderer Informationen, die mit den eigenen Daten verbunden sind, möglich. Ähnlich dem klassischen Web of Hypertext ist das Web of Data auf Dokumenten aufgebaut. Im Gegensatz zum Web of Hypertext, in dem Links Beziehungen über HTML[76] eingehen, werden die Links im Web of Data durch RDF[77] beschrieben [BERNERS-LEE, 2009]. Durch einen Uniform Resource Identifier[78] (URI) kann eine kompakte Folge von Zeichen, die eine abstrakte oder physische Ressource repräsentieren, im World Wide World angesprochen werden. Sie besteht aus fünf Teilen: scheme, autority, path, query und einem fragment, vgl. Abbildung 2-8 [Berners-Lee et al., 2005].

```
URI         = scheme ":" hier-part [ "?" query ] [ "#" fragment ]

hier-part   = "//" authority path-abempty
            / path-absolute
            / path-rootless
            / path-empty

  foo://example.com:8042/over/there?name=ferret#nose
  \_/   \______________/\_________/ \_________/ \__/
   |           |            |            |        |
scheme     authority       path        query   fragment
   |   _____________________|__
  / \ /                     \
  urn:example:animal:ferret:nose
```

Abbildung 2-8: Aufbau einer URI [Berners-Lee et al., 2005]

Nachfolgend sind einige Beispiele für URIs gegeben:

- http://de.wikipedia.org/wiki/Mainz
- ftp://ftp.is.co.za/rfc/rfc1808.txt
- file:///C:/tmp/Masterarbeit.docx
- mailto:florian.thiery@geoinform.fh-mainz.de

Ein URI kann somit jede Art von Objekten oder Konzepten identifizieren. Diese Idee soll auch auf HTML und RDF übertragen werden. Dies führt zu den Grundprinzipien, bzw. Regeln der Linked Data (vgl. BERNERS-LEE, 2009):

1. "Use URIs as names for things."
2. "Use HTTP URIs so that people can look up those names."
3. "When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)"
4. "Include links to other URIs. so that they can discover more things."

---

[74] siehe auch http://linkeddata.org (abgerufen 03.12.2013)

[75] vgl. http://www.w3.org/People/Berners-Lee (abgerufen 03.12.2013)

[76] vgl. http://www.w3.org/html (abgerufen 03.12.2013)

[77] vgl. http://www.w3.org/RDF (abgerufen 03.12.2013)

[78] vgl. http://tools.ietf.org/html/rfc3986 (abgerufen 03.12.2013)

Regel eins fordert für die Bezeichnung von Objekten, wie Web-Dokumente, digitale Inhalte, aber auch Objekte der realen Welt und abstrakte Konzepte, eine URI. Objekte können jedoch konkrete Dinge wie Personen, Orte und Autos oder abstrakte Ausprägungen, wie persönliche Beziehungen, ein Datensatz aller grünen Autos der Welt, bzw. die Farbe Grün selbst, sein. Das Prinzip kann als Erweiterung des Anwendungsbereichs des Webs und Online Ressourcen gesehen werden, sodass ein beliebiges Objekt oder Konzept der Welt modelliert werden kann. Wird eine URI, inklusive einem universellen Satzes von Symbolen[79], nicht verwendet, kann nicht von Semantic Web gesprochen werden. Das HTTP Protokoll[80] ist der universelle Zugriffsmechanismus des Webs. Im klassischen Web werden HTTP-URIs verwendet. Mit Hilfe dieser, können global eindeutige Bezeichner mit einfachen Retrieval-Methoden[81] durchsucht werden. Die zweite Regel befürwortet den Einsatz von HTTP-URIs, so dass die zuvor erwähnten Vorteile genutzt werden können. Verschiedenste Anwendungen zur Verarbeitung von Webinhalten erfordern standardisierte Formate. Die Vereinbarung zur Nutzung von HTML als dominierendes Austauschformat in World Wide Web war ein wichtiger Faktor, den das WWW so erfolgreich gemacht hat. Die dritte Regel spricht sich daher für ein einziges Datenmodell zur Veröffentlichung von strukturierten Daten im Web aus: Das Ressource Description Framwork (RDF), vgl. auch Kapitel 2.5.3. RDF basiert auf einem einfachen Graphenmodell[82]. Als Basisformat kann RDF/XML[83] (oder dessen Dialekt Turtle[84]) angesehen werden. Eine Abfrage dieser Daten sollte über eine Abfragesprache, hier SPARQL[85] (vgl. Kapitel 2.5.4) ermöglicht werden. Regel vier zeigt das große Potential der Linked Data. Durch Links in den eigenen modellierten Daten zu anderen im Web verfügbaren Repositorien entsteht ein Web of Data. Im Web of Hypertext ist es ein unausgesprochener Standard, dass Links zu externen, weiterführenden Inhalten erzeugt werden. Der Wert der eigenen Information hängt so maßgeblich auch von den verlinkten Inhalten ab. Dieses Prinzip kann direkt auf das Semantic Web und Linked Data übertragen werden (vgl. BERNERS-LEE, 2009 und HEATH ET AL., 2011). Weiterführende Informationen sind ebenfalls BERNERS-LEE (2009) und HEATH ET AL. (2011) zu entnehmen.

Abbildung 2-9 zeigt das Linking Open Data cloud diagram (Stand September 2011), das publizierte Linked Data Datensätze und deren Verlinkungen beschreibt (LODCLOUD, 2013A). Zusammenfassend ist zu sagen, dass Linked Data unerlässlich sind, damit Verbindungen des Semantic Web beschreiben werden können. Vorschläge, in welcher Art und Weise Linked Data im Web publiziert werden sollten, macht unter Anderem BIZER ET AL. (2007).

---

[79] siehe auch http://en.wikipedia.org/wiki/Percent-encoding (abgerufen 03.12.2013)

[80] vgl. http://www.w3.org/Protocols (abgerufen 03.12.2013)

[81] siehe auch http://de.wikipedia.org/wiki/Information_Retrieval (abgerufen 03.12.2013)

[82] siehe auch http://www.w3.org/TR/rdf-concepts

[83] vgl. http://www.w3.org/TR/REC-rdf-syntax (abgerufen 03.12.2013)

[84] vgl. http://www.w3.org/TR/turtle (abgerufen 03.12.2013)

[85] vgl. http://www.w3.org/TR/rdf-sparql-query (abgerufen 03.12.2013)

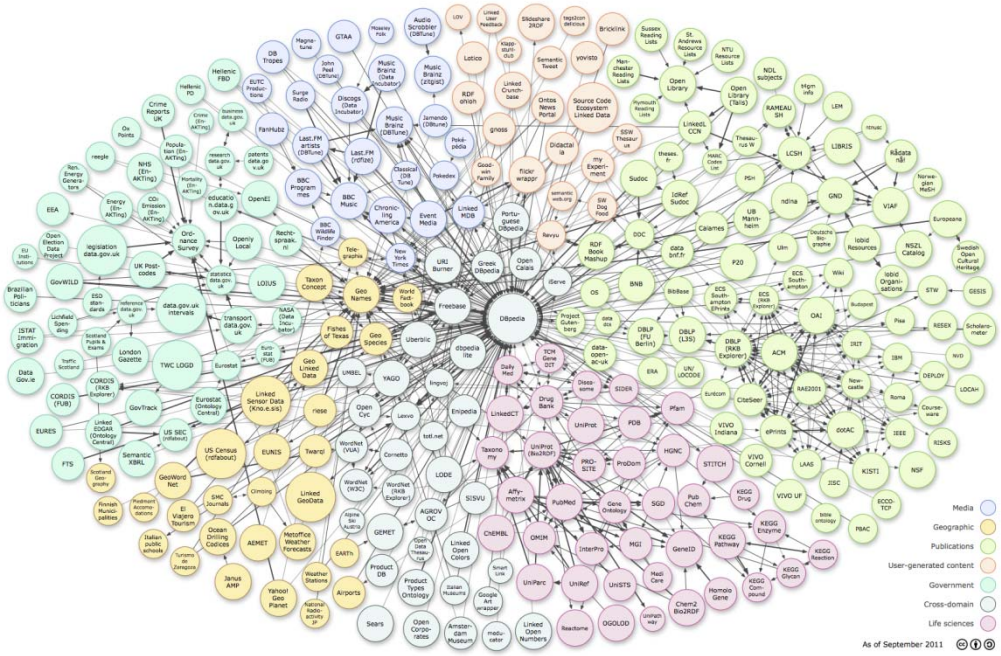


Abbildung 2-9: Linked Data Cloud [LODCLOUD, 2013B]

Der bereits zuvor erwähnte Tim Berners Lee (Begründer des WWW und Direktor des W3C Konsortiums) hat ein sogenanntes 5-Sterne-Modell[86] (vgl. Abbildung 2-10) für Open Government Data[87] entwickelt. Ziel dieses Modells ist die Entwicklung einer Linked- Government-Data-Infrastruktur, die auf Basis von offenen W3C Standards Linked Open Data (LOD) anbieten. Diese Maßnahme hat für die digitale Infrastruktur eines Landes den großen Vorteil der Interoperabilität. Die Veröffentlichung (nicht nur von Regierungsdaten, sondern auch anderen Daten öffentlicher Träger und Institutionen) von Daten zu einer Wiederverwendung durch andere und Verknüpfung dieser mir weiteren externen Datenbeständen, baut den Vorteil der Interoperabilität weiter aus.

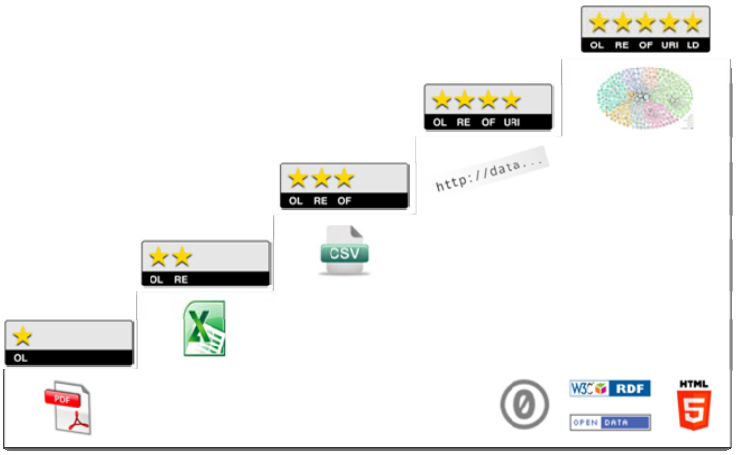


Abbildung 2-10: 5 Star Data Überblick [5STARDATA, 2013]

[86] vgl. http://5stardata.info (abgerufen 03.12.2013)

[87] siehe auch (abgerufen 03.12.2013)
http://www.zu.de/deutsch/lehrstuehle/ticc/TICC-101203-OpenGovernmentData-V1.pdf

Verschiedene Stufen des Modells (Sterne) zeigen bestimmte Vorteile des Nutzers und des Datenhalters (vgl. Abbildung 2-11). Eine Besonderheit dieser Stufen ist, dass alle Eigenschaften der niedrigeren Stufen von den höheren Stufen geerbt werden. Daten, die im Netz vorliegen, jedoch keine offene Lizenz[88] zur Nutzung dieser Daten anbieten (z.B. handgeschriebene Tabelle auf einem Blatt Papier), erhalten in Berners-Lees Skala keinen Stern (Web Data: WO[89]). Liegen die Daten im Web (Format egal) mit einer offenen Lizenz vor (z.B. eingescannte handgeschriebene Tabelle), so kann ein Stern (Open Licence: OL) vergeben werden. Diese Daten können von einem Nutzer z.B. angesehen, ausgedruckt und in andere manuelle Formate übertragen werden. Die Veröffentlichung durch den Datenhalter ist einfach. Eine Steigerung stellt die Veröffentlichung in einem strukturierten Format (z.B. Speicherung der Tabelle in Excel) dar: 2 Sterne (Reused: RE). Die Daten können mit proprietärer Software weiterverarbeitet und in andere Formate übertragen werden. Liegen die Daten in einem strukturierten, jedoch nicht proprietärem Format (z.B. Darstellung der Tabelle als CSV[90]) vor, werden drei Sterne vergeben (OF: Open Formats). Die Bearbeitung der Daten ohne proprietäre Software ist möglich, der Datenhalter muss jedoch gegebenenfalls auf Konverter zurückgreifen, die beispielsweise aus einem Excel-File eine CSV-Datei erstellen. Werden die Daten unter einer eindeutigen URL (URI) zur Verfügung gestellt, ist es möglich, Verbindungen mit anderen Ressourcen herzustellen: Stufe 4 (URI). In diesem Fall muss der Datenhalter, die Daten strukturell aufarbeiten und URIs für diese vergeben. Die höchste Stufe und somit die 5-Sterne-Kategorie (LD: Linked Data), ist die Verlinkung der eigenen Daten mit anderen Daten, zur Erstellung eines Kontextes. Hierbei kann ein Nutzer unter anderem ein Datenschema entdecken (z.B. XML oder RDF), das er zur weiteren Verarbeitung nutzen kann. Der Datenhalter benötigt in diesem Fall Ressourcen (URIs), die Links herstellen und verbessert so den Zugriff von Suchmaschinen auf die eigenen Daten.

---

[88] siehe auch http://de.creativecommons.org (abgerufen 03.12.2013)

[89] Erläuterung der Abkürzungen vgl. http://lab.linkeddata.deri.ie/2010/lod-badges (abgerufen 03.12.2013)

[90] vgl. http://tools.ietf.org/html/rfc4180 (abgerufen 03.12.2013)

Kein Stern	WD	Daten im Web (Format egal), ohne offene Lizenz		
Ein Stern	OL	Daten im Web (Format egal) mit offener Lizenz	Der Nutzer kann ... • die Daten ansehen • die Daten Ausdrucken • die Daten lokal speichern • die Daten manuell in andere Formate übertragen	Der Datenhalter kann ... • einfach veröffentlichen
Zwei Sterne	OL RE	Daten in strukturiertem Format (z.B. Excel)	Der Nutzer kann ... • alles was er mit 1-Stern-Daten machen kann • die Daten mit properitärer Software weiterverarbeiten • die Daten in andere Formate übertragen	Der Datenhalter kann ... • einfach veröffentlichen
Drei Sterne	OL RE OF	Daten in strukturiertem, nicht proprietärem Format (z.B. CSV statt Excel)	Der Nutzer kann ... • alles was er mit 2-Stern-Daten machen kann • die Daten mit weiterverarbeiten, ohne auf properitäre Software angewiesen zu sein	Der Datenhalter ... • benötigt möglicherweise Konverter oder Konnektoren um Daten aus seinen properitären Formaten bereitzustellen
Vier Sterne	OL RE OF URI	Verwendung von eindeutigen URLs, so dass Datensätze verlinkt werden können	Der Nutzer kann ... • alles was er mit 3-Stern-Daten machen kann • Verbindungen mit anderen Ressourcen herstellen • Lesezeichen auf Datensätze legen	Der Datenhalter ... • bereitet Daten und Datensätze strukturell auf • vergibt URIs für Daten • bekommt Daten über Zugriffe und kann dementsprechend die Datenpräsenz optimieren
Fünf Sterne	OL RE OF URI LD	Verlinkung der eigenen Daten mit anderen Daten, um Kontext herzustellen	Der Nutzer kann ... • alles was er mit 4-Stern-Daten machen kann • Daten wie Hypertext verwenden, indem ein Datum ein nächstes verlinked • das Datenschema entdecken und nutzen	Der Datenhalter ... • benötigt Ressourcen um die Links herzustellen • macht seine Daten für Suchmaschinen effektiv auffindbar • steigert den Verwendungswert der Daten

Abbildung 2-11: 5 Star Data Eigenschaften [OSW, 2013]

Im Rahmen dieser Arbeit wird der Status der vier- bzw. fünf-Sterne-Daten angestrebt (vgl. Abbildung 2-12). Die Daten liegen zunächst in einer Access Datenbank (0 Sterne) bzw. in einer nicht frei verfügbaren Onlineplattform[91] ohne eindeutige Bezeichner (URI) vor (ebenfalls 0 Sterne). Die Daten werden nach einer Migration mit Hilfe der ReST-Schnittstelle (vgl. Kapitel 6.1) in nicht proprietären Formaten, wie XML, JSON und RDF, als URI zur Verfügung gestellt (4 Sterne). In diesen Daten werden Links zu weiteren Ressourcen (z.B. Keramikfragment zu Töpfer oder Findspot zu einem Pleiades Place) erzeugt und somit die Möglichkeit gegeben, neue Informationen, auch aus anderen Repositorien, zu beziehen (5 Sterne).

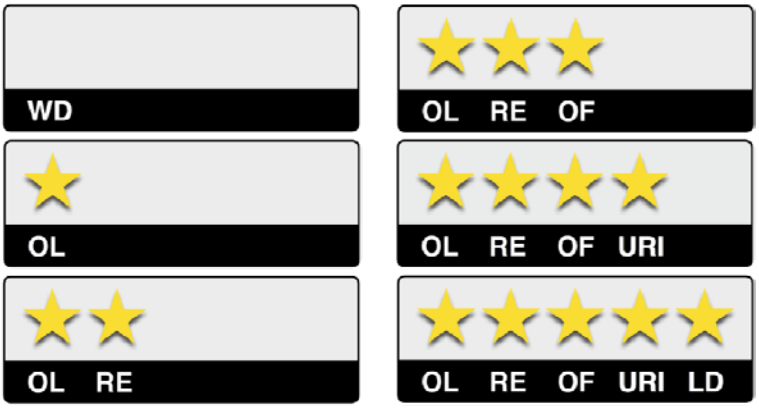


Abbildung 2-12: 5 Star Data Model Sterne

[91] vgl. http://www.rgzm.de/samian (abgerufen 03.12.2013)

2.5.3 Ressource Description Framework (RDF)

Das Resource Description Framework (RDF) ist die Sprache des Semantic Web (vgl. Kapitel 2.5.1) und der Linked Data (vgl. Kapitel 2.5.2). Die nachfolgenden Ausführungen und Beispiele entstammen HITZLER ET AL. (2008) und HAFT (2013) bzw. sind diesen nachempfunden.

Die Beschreibung strukturierter Informationen (z.B. auch Ressourcen) kann mit Hilfe einer formalisierten Sprache erfolgen. Eine dieser Sprachen ist RDF[92]. RDF ermöglicht einen Datenaustausch über das Web, ohne dass dabei die ursprünglichen Bedeutungen verloren gehen. Insbesondere die Kombination und Weiterverarbeitung der enthaltenen Informationen zeigt einen Gegensatz zu bekannten Formaten wie XML und HTML, in denen die Darstellung von Dokumenten im Vordergrund steht. Heutzutage können eine Vielzahl an praktischen Werkzeugen zum Umgang mit RDF abgerufen werden, wie auch Systeme zur Auswertung dieser großen Menge an RDF-Daten (z.B. Triple Stores). In einigen Anwendungsgebieten wird RDF schon als Austausch von (Meta-)Daten eingesetzt, z.B. das Format RSS 1.0[93] zum Abonnement von Nachrichten [HITZLER ET AL., 2008].

RDF-Dokumente beschreiben gerichtete Graphen[94] und somit eine Menge von Knoten, die durch gerichtete Kanten (zeichnerisch auch Pfeile) miteinander verbunden sind. Abbildung 2-13 zeigt ein einfaches Beispiel eines Graphen mit zwei Knoten (Balbinus, Langenhain) und einer Kante (arbeitetIn), das einer Person einen Arbeitsplatz zuweist. In RDF werden Knoten und Kanten immer mit eindeutigen Bezeichnern (hier z.B. http://toepfer.de/Balbinus) beschriftet, so dass Verwechslungen ausgeschlossen werden können (vgl. Abbildung 2-14). Im Vergleich zu RDF-Dokumenten werden XML- Dokumente in einer Baumstruktur abgelegt. Die Organisation von Informationen in elektronischen Dokumenten ist meist hierarchisch organisiert, was sich mit Bäumen optimal ausdrücken lässt. RDF ist nicht für eine hierarchische Entwicklung einzelner Dokumente konzipiert, sondern für eine Beschreibung von Beziehungen zwischen Ressourcen. Möchte man beispielsweise ausdrücken, dass der Töpfer in einer Werkstatt in Langenhain arbeitet, so ist die Beziehung zwischen Töpfer und Werkstatt eine Information, die keiner der beiden Ressourcen untergeordnet ist. Aus solchen Beziehungen entstehen natürlicherweise Graphen, jedoch keine Baumstrukturen. Die dezentrale Verwaltung von Informationen im WWW ist ein weiterer Grund zur Repräsentation als Graph. Mit RDF ist es kein Problem, Informationen aus verschiedensten Quellen zu kombinieren. Durch diese Möglichkeit können größere Graphen entstehen, die weitere neue Informationen zur Auswertung liefern können. Eine Bereitstellung von XML-Dokumenten im Web und das Auslesen von Informationen aus diesen Baumstrukturen wären wesentlich komplizierter. Letztendlich lässt sich sagen, dass Graphen zur Komposition dezentraler Informationen geeignet sind [HITZLER ET AL., 2008].

---

[92] vgl. http://www.w3.org/RDF (abgerufen 03.12.2013)

[93] siehe auch http://web.resource.org/rss/1.0/spec (abgerufen 03.12.2013)

[94] siehe auch http://de.wikipedia.org/wiki/Graph_(Graphentheorie) (abgerufen 03.12.2013)

Abbildung 2-13: Graph Beispiel

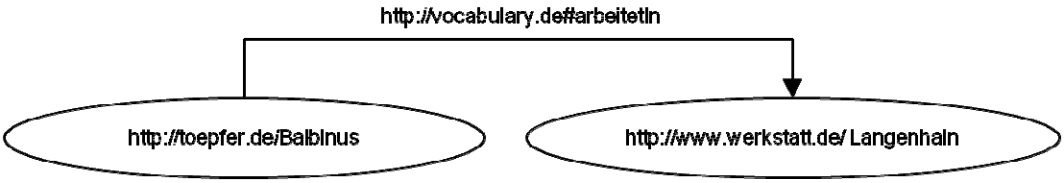


Abbildung 2-14: RDF-Graph Beispiel mit URIs

RDF Graphen werden als Kompositionen von verteilten Daten betrachtet. Ein wesentliches Problem hierbei ist, dass Ressourcen in verschiedenen RDF-Dokumenten verschiedene Bezeichner haben können. So kann es vorkommen, dass gleiche Ressourcen unterschiedlich benannt werden, oder dass für den gleichen Bezeichner unterschiedliche Ressourcen gebraucht werden. Das zweite erwähnte Problem kann mit Hilfe von Uniform Ressource Identifiern (URIs) gelöst werden. URIs sind eine Verallgemeinerung von URLs und beschreiben Webadressen, die den Zugriff auf ein Online-Dokument ermöglichen. Auch wenn Ressourcen nicht online abgerufen werden können (z.B. das abstrakte Konzept eines Töpfers) verwendet man URIs, so ist eine Verwechslung von Ressourcen auszuschließen, siehe auch Abbildung 2-14. Ausgenommen davon sind lediglich Literale (Text oder Zahlenwerte) und so genannte leere Knoten. Auf beide Themen wird diesem Kapitel noch verwiesen [HITZLER ET AL., 2008]. Betrachtet man den Aufbau einer URL bzw. URI, stellt sich die Frage nach dem korrekten Namespace (Namensraum). Um sicherzustellen, dass ein URI nicht schon einer anderen Ressource zugeordnet ist, ist die eigene Domain zu empfehlen [HAFT, 2013]. Für diese Masterarbeit wurde noch keine DNS-Domain registriert, so dass der Namespace als provisorisch angesehen werden kann: http://143.93.114.104/rest/. Eine Beschreibung von Ressourcen und Vokabularien kann letzten Endes z.B. unter einer Domain wie http://samianware.net/rest/ erfolgen.

Mit URIs können (abstrakte) Ressourcen beschrieben werden. URIs werden dabei lediglich als Referenzen auf tatsächlich gemeinte Dinge gesehen und bieten gegeben falls als im Web adressierbare Adresse, weitere Informationen in RDF-Notation an. Oftmals soll die Interpretation des Bezeichners nicht von Sonderfunktionen einzelner Anwendungen abhängen, sondern auf einzelne gezielte Werte beschränkt sein. So können Datenwerte wie Zahlen, Namen, Wahrheits- oder Datumswerte ebenfalls adressiert werden. In diesem Fall werden so genannte Literale genutzt. Literale sind reservierte Bezeichner, die RDF Ressourcen bestimmter Datentypen darstellen. Jeder Wert eines Literals wird allgemein durch eine Zeichenkette beschrieben, z.B. „42", „Langenhain" oder „true". Literale, für die kein expliziter Datentyp angegeben ist, werden als ungetyptes Literal bezeichnet. Litreale können in RDF nie Ausgangspunkt von Kanten in einem RDF-Graph sein, so dass man keine direkten Aussagen über Literale tätigen kann. Zudem können Kanten ebenfalls nicht mit Literalen beschriftet werden [HITZLER ET AL., 2008]. Ein Beispiel für ein ungetyptes

Literal ist in Abbildung 2-15 aufgezeigt. Hierbei wird der Ressource Werkstatt Langenhain das Literal Balbinus zugewiesen.

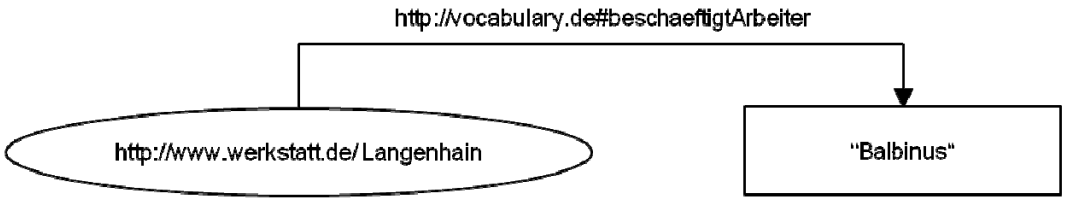


Abbildung 2-15: RDF ungetyptes Literal

Damit Verwechslungen (z.B. 42 als Zahl oder „42" als Text) bei Literalen ausgeschlossen werden können, sind Typisierungen vorzusehen. RDF empfiehlt hierbei die Verwendung von Datentypen des XML Schema[95], da es selbst keine Datentypen kennt [HITZLER ET AL., 2008]. Abbildung 2-16 zeigt das zuvor erwähnte Beispiel mit einen als String getypten Literal:

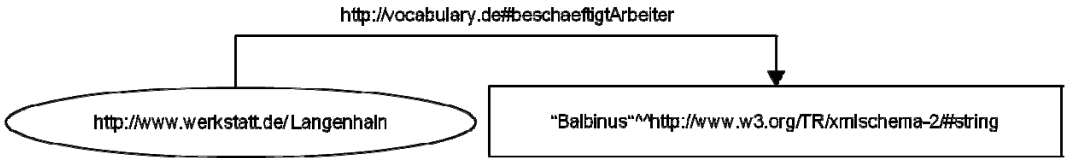


Abbildung 2-16: RDF getyptes Literal

Die bereits gezeigten Beispiele stellen (einfache) RDF Graphen in einer Grafik dar. Diese sind für eine Verarbeitung der Daten mit Hilfe von Computersystemen ungeeignet. Wird eine höhere Komplexität in den Datenverknüpfungen erreicht, ist zudem die Lesbarkeit für Menschen ebenfalls nur noch bedingt möglich. Zerlegt man den Graphen in seine Bestandteile und speichert sie der Reihe nach ab („Umwandlung komplexer Datenobjekte in lineare Zeichenketten"), wird dies als Serialisierung bezeichnet. Ein Graph kann in der Informatik beispielsweise durch eine Erreichbarkeitsmatrix dargestellt werden (siehe z.B. die Adjazenzmatrix[96] des Dijkstra-Algorithmus[97]). RDF-Graphen sind jedoch lichte Graphen, in denen die meisten denkbaren Beziehungen nicht gelten. In diesem Fall kann es als sinnvoll erachtet werden, dass jede Knoten-Kanten-Beziehung einzeln abgespeichert wird. Abbildung 2-17 zeigt genau eine Kante, die durch ihren Anfangs- und Endpunkt eindeutig bezeichnet wird. Diese drei Bestimmungsstücke werden als Subjekt (subject), Prädikat (predicate) und Objekt (object) bezeichnet. Jeder Graph kann vollständig durch die Angaben seiner Kanten beschrieben werden. Sie entsprechen immer einem so genannten Tripel 'Subject-Prädikat-Objekt'. Dies bedingt, dass die zuvor beschriebenen Literale nur als Objekt auftreten können [HITZLER ET AL., 2008].

---

[95] siehe auch http://www.w3.org/TR/xmlschema-2 (abgerufen 03.12.2013)

[96] siehe auch http://de.wikipedia.org/wiki/Adjazenzmatrix (abgerufen 03.12.2013)

[97] siehe auch http://de.wikipedia.org/wiki/Dijkstra-Algorithmus (abgerufen 03.12.2013)

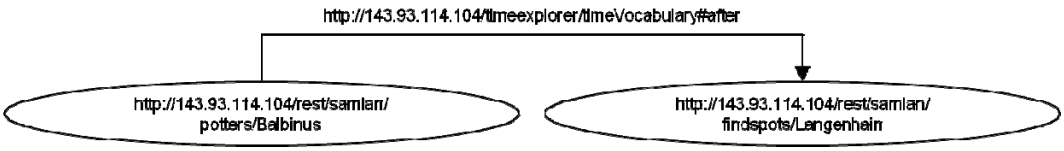


Abbildung 2-17: Einfacher RDF Graph

Eine Darstellung von Triple-Listen ist in verschiedenen Formaten möglich. Die gängigsten Formate sind, das auf XML aufbauende RDF/XML[98], sowie die für Menschen einfacher lesbare Turtle[99] (Terse RDF Triple Language) Notation. Viele Bibliotheken (auch das in dieser Arbeit verwendete Apache Jena Framework) nutzen das komplexere RDF/XML, da eine XML Repräsentation in der Informatik geläufiger ist. Als inoffizieller Standard gilt der RDF-Turtle-Dialekt [HAFT, 2013]. Der in Abbildung 2-17 gezeigte Graph würde nach einer Serialisierung in RDF/XML[100] folgendem Dokument gleichen (siehe Abbildung 2-18):

```
1 <?xml version="1.0"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" xmlns:timeVocabulary="http://143.93.114.104/timeexplorer/timeVocabulary#">
3         <rdf:Description rdf:about="http://143.93.114.104/rest/samian/potters/Balbinus">
4                 <timeVocabulary:after rdf:resource="http://143.93.114.104/rest/samian/findspots/Langenhain" />
5         </rdf:Description>
6 </rdf:RDF>
```

Abbildung 2-18: RDF/XML Beispiel

Der gleiche Graph kann auch in Turtle-Notation modelliert werden:

```
1 <http://143.93.114.104/rest/samian/potters/Balbinus>
  <http://143.93.114.104/timeexplorer/timeVocabulary#after>
  <http://143.93.114.104/rest/samian/findspots/Langenhain> .
```

Beschreibt man einen Graphen in Turtle-Syntax, werden Subjekt, Prädikat und Objekt jeweils durch ein Leerzeichen getrennt, hintereinander geschrieben, URIs in spitze Klammern eingeschlossen und das Tripel durch einen Punkt am Ende der Zeile abgeschlossen. Die Übersichtlichkeit in diesen Dokumenten kann durch eine Einführung von Namensräumen, sogenannter Präfixe, erhöht werden. URIs werden in diesem Fall abgekürzt [HAFT, 2013]. Das zuvor beschriebene Turtle Dokument kann gekürzt wie folgt geschrieben werden:

```
1 @prefix potter: <http://143.93.114.104/rest/samian/potters/> .
2 @prefix pred: <http://143.93.114.104/timeexplorer/timeVocabulary#> .
3 @prefix findspot: <http://143.93.114.104/rest/samian/findspots/> .
4
5 potter:Balbinus pred:after findspot:Langenhain .
```

[98] vgl. http://www.w3.org/TR/rdf-syntax-grammar (abgerufen 03.12.2013)

[99] vgl. http://www.w3.org/TeamSubmission/turtle (abgerufen 03.12.2013)

[100] weitere Informationen zur Darstellung in RDF/XML sind HITZLER ET AL. (2008) zu entnehmen

Zeile 5 zeigt das eigentliche Tripel. In den Zeilen 1 bis 3 werden die Präfixe definiert. Diese sind frei wählbar, einige bekannte Präfixe sind jedoch zu verwenden (z.B: dcterms, rdf, oder rdfs). Der Töpfer Potter wird somit an das letzte Element des Präfixes in der Klammer angehängt, vgl. http://143.93.114.104/rest/samian/potters/Balbinus. Liegen mehrere Tripel desselben Subjekts, bzw. desselben Subjekts und Prädikats vor, (vgl. Abbildung 2-19) kann eine Kurzform verwendet werden.

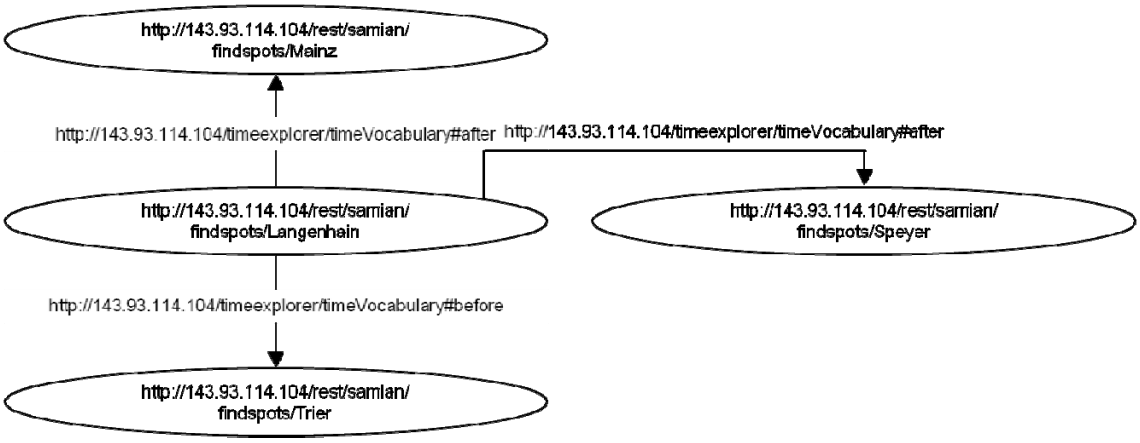


Abbildung 2-19: Graph mit identischem Subjekt

Eine Serialisierung des oben gezeigten Graphen (inkl. verschiedener Präfixe) würde z.B. folgendes Resultat ergeben:

```
1 @prefix pred: <http://143.93.114.104/timeexplorer/timeVocabulary#> .
2 @prefix findspot: <http://143.93.114.104/rest/samian/findspots/> .
3
4 findspot:Langenhain pred:after findspot:Mainz .
5 findspot:Langenhain pred:before findspot:Trier .
6 findspot:Langenhain pred:after findspot:Speyer .
```

Es zeigt sich, dass hier mehrfach das Gleiche Subjekt (Langenhain) und zweimal das gleiche Prädikat (after) verwendet wird. Dieser Fall kann mit Kurzschreibweisen gelöst werden. Hierbei trennt ein Semikolon Tripel desselben Subjekts und Kommata Tripel, die sich das Subjekt und Prädikat teilen. Dieser Struktur wird ebenfalls durch einen Punkt abgeschlossen, siehe nachfolgendes Turtle-Dokument.

```
1 @prefix pred: <http://143.93.114.104/timeexplorer/timeVocabulary#> .
2 @prefix findspot: <http://143.93.114.104/rest/samian/findspots/> .
3
4 findspot:Langenhain pred:before findspot:Trier ;
5                     pred:after  findspot:Mainz ,
6                                 findspot:Speyer .
```

In dieser Arbeit werden sowohl RDF/XML, als auch die Turtle Notation genutzt. Kurzschreibweisen werden aufgrund der Übersichtlichkeit ebenfalls eingeführt[101]. Ein weiteres Phänomen stellen Blank Nodes, so genannte leere Knoten dar. Hierzu sein ein Beispielgraph gegeben (vgl. Abbildung 2-20).

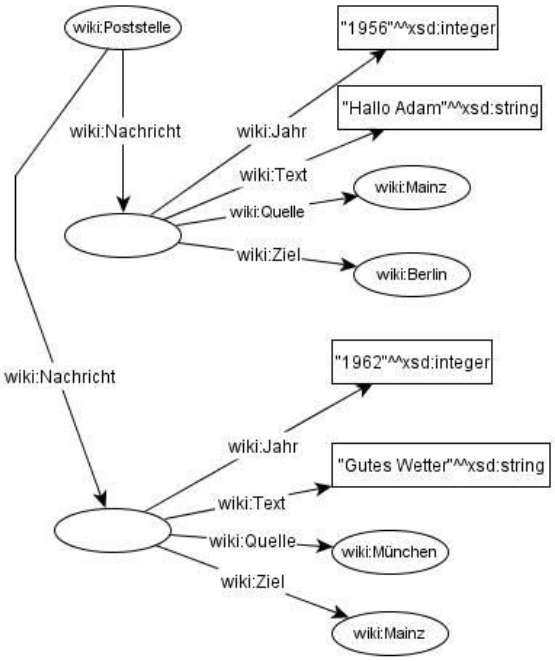


Abbildung 2-20: Beispiel leerer Knoten

Hier ist zu erkennen, dass eine Nachricht viele von ihr abhängende Eigenschaften enthält. Die Nachricht kann als mehrwertiger Bezeichner angesehen werden. Somit existiert nicht die EINE Nachricht, die mit einem URI eindeutig adressierbar ist, sondern viele unterschiedliche Ausprägungen dieser. Eine sinnvolle Abbildung in RDF wird hierbei über leere Knoten realisiert. Sie sind Hilfsknoten, die mit Eigenschaftswerten verbunden sind. Weitere Nachrichten können mit einem weiteren leeren Knoten hinzugefügt werden. Ein leerer Knoten wird in Turtle Syntax durch eine ID (z.B. _:id) gekennzeichnet, vgl. nachfolgendes Dokument.

[101] siehe z.B. http://143.93.114.104/rest/samian/findspots/Langenhain.ttl (abgerufen 03.12.2013)

```
1  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
2  @prefix wiki: <https://de.wikipedia.org/wiki/> .
3
4  wiki:Postamt wiki:Nachricht _:id1 .
5  _:id1        wiki:Jahr      "1956"^^xsd:integer ;
6               wiki:Text      "Hallo Adam"^^xsd:string ;
7               wiki:Quelle    wiki:Mainz ;
8               wiki:Ziel      wiki:Berlin.
9  wiki:Postamt wiki:Nachricht _:id2 .
10 _:id2        wiki:Jahr      "1962"^^xsd:integer ;
11              wiki:Text      "Gutes wetter"^^xsd:string ;
12              wiki:Quelle    wiki:Muenchen ;
13              wiki:Ziel      wiki:Mainz.
```

Nachfolgend ist ein weiteres Beispiel eines RDF-Graphen (hier eines Stammbaums) gezeigt. Zunächst werden (einfache) Sätze formuliert, die mit Hilfe einer Grafik (vgl. Abbildung 2-21) in Turtle Notation übertragen werden. Des Weiteren wird auch die Möglichkeit von SPARQL-Abfragen (vgl. Kapitel 2.5.4) und damit der Generierung neuen Wissens, hier Enkel, bzw. Geschwister vorgestellt. Auch die Modellierung eines Würfels (und damit geometrische Beziehungen) und diverse Abfragen, ist mit RDF möglich. Dieses Beispiel ist Anhang A: RDF Beispiel Cube zu entnehmen.

Sachverhalt schriftlich (Stammbaum)

- Anton ist Vater von Christine
- Berta ist Mutter von Christine
- Anton ist Vater von Daniel
- Berta ist Mutter von Daniel
- Emil ist Vater von Friedrich
- Christine ist Mutter von Friedrich
- Daniel ist Vater von Gustav
- Heike ist Mutter von Gustav

Sachverhalt (grafisch)

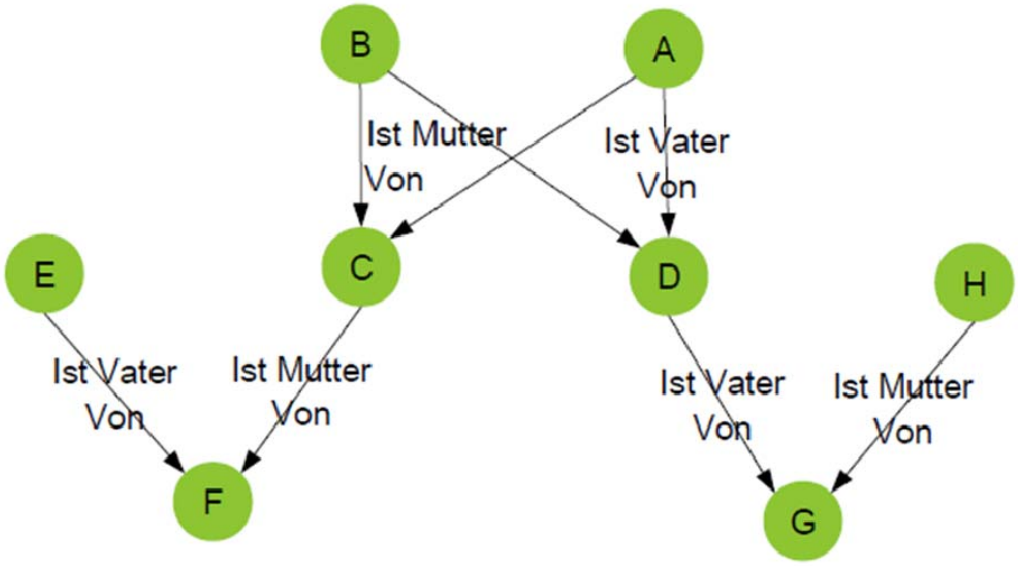


Abbildung 2-21: RDF Beispiel (Stammbaum)

Sachverhalt (Turtle-Syntax)

```
@prefix person:    <http://www.beispiel.de/person/> .
@prefix voc:       <http://www.beispiel.de/vokabular/> .
person:Anton       voc:istVaterVon      person:Christine .
person:Berta       voc:istMutterVon     person:Christine .
person:Anton       voc:istVaterVon      person:Daniel .
person:Berta       voc:istMutterVon     person:Daniel .
person:Emil        voc:istVaterVon      person:Friedrich .
person:Christine   voc:istMutterVon     person:Friedrich .
person:Daniel      voc:istVaterVon      person:Gustav .
person:Heike       voc:istMutterVon     person:Gustav .
```

Beispiel-Abfragen

1. Wer ist Opa und von wem?

```
1 PREFIX person: <http://www.beispiel.de/person/>
2 PREFIX voc: <http://www.beispiel.de/vokabular/>
3 SELECT ?opa ?enkel
4 WHERE {
5  ?opa voc:istVaterVon ?x .
6  { ?x voc:istVaterVon ?enkel . }
7  UNION
8  { ?x voc:istMutterVon ?enkel . }
9 }
```

2. Wer sind Geschwister?

```
1 PREFIX person: <http://www.beispiel.de/person/>
2 PREFIX voc: <http://www.beispiel.de/vokabular/>
3 SELECT ?kind1 ?kind2
4 WHERE {
5  ?v voc:istVaterVon ?kind1 ,
6  ?kind2 .
7  ?m voc:istMutterVon ?kind1 ,
8  ?kind2 .
9 }
```

2.5.4 SPARQL

SPARQL[102] (SPARQL Protocol and RDF Query Language) ist ein relativ neuer Standard[103] und eine Anfrage- bzw. Abfragesprache für RDF-Graphen und die Darstellung der Resultate. SPARQL enthält zudem erweiterte Funktionen für die Konstruktion komplexer Anfragen, wie Filterbedingungen und die Formatierung der Ausgabe. Nachfolgend wird nur auf die SPARQL-Anfragesprache

[102] vgl. http://www.w3.org/TR/sparql11-query/ (abgerufen 03.12.2013)

[103] zur Zeit noch eine Recommendation (d h. die W3C Arbeitsgruppe hat ein vorläufiges Endergebnis vorgelegt)

eingegangen, das SPARQL-Protokoll und die SPARQL-Kodierung werden nicht weiter behandelt [HITZLER ET AL., 2008].

In SPARQL werden einfache RDF-Graphen als Anfragemuster verwendet, da sich SPARQL selbst als Anfragesprache für (RDF-)Graphen sieht. Die Turtle-Syntax dient dabei als Grundlage des Anfragegraphen. SPARQL führt zudem weitere Anfragevariablen ein, die dazu verwendet werden können, bestimmte Elemente im Anfragegraphen als Ergebnis zu ermitteln. Eine Anfrage besteht aus drei wesentlichen Teilen: PREFIX, SELECT und WHERE [HITZLER ET AL., 2008].

```
1 PREFIX potter: <http://143.93.114.104/rest/samian/potters/>
2 PREFIX samian: <http://143.93.114.104/rest/samian/>
3 SELECT ?name ?geschlecht
4 WHERE {
5   ?potter samian:Werkstatt <http://www.potterwerkstatt.de/001> .
6   ?potter samian:Name ?name .
7   ?potter samian:Geschlecht ?geschlecht .
8 }
```

Zeile 1 zeigt das Schlüsselwort PREFIX, ähnlich der @prefix Notation der Turtle-Syntax. Jedoch ist hier kein abschließender Punkt am Ende Zeile nötig. Ein weiteres Schlüsselwort ist in Zeile 2 auffindbar: SELECT. Es bestimmt im Allgemeinen das Ausgabeformat. Hier können sowohl Werte konkreter Variablen (gekennzeichnet durch ein Fragezeichen), als auch eine 'Wildcard' (z.B. Select *), das heißt alle angefragten Variablen des WHERE Ausdrucks, angegeben werden. In diesem Beispiel werden alle Rückgabewerte der Variablen ?name und ?geschlecht angezeigt, die Variable ?potter hingehen, bleibt unberührt. Das letzte Schlüsselwort WHERE leitet die eigentliche Anfrage ein. Eingeschlossen in geschweifte Klammern folgt an dieser Stelle ein einfaches Graph-Muster. Im Fall des gezeigten Beispiels besteht dies aus drei Tripeln in RDF Turtle Dialekt. Im Vergleich zur Turtle-Notation jedoch treten hier nicht nur URIs (z.B. http://www.potterwerkstatt.de/) sondern auch Variablenbezeichner (z.B. ?potter) auf. Die Variablen repräsentieren eine Art Platzhalter, welche erst durch eine Anfrage mit konkreten Werten gefüllt werden können. Tritt eine Variable mehrfach auf, so heißt dies, dass an diesen Stellen das gleiche Element vorhanden sein muss [HITZLER ET AL., 2008].

Die obenstehende Anfrage sucht somit nach allen Töpfern, die in der Töpferwerkstadt 001 arbeiten und mit einem Namen und ihrem Geschlecht beschrieben sind. Zur Verdeutlichung sei nachfolgend beispielhaftes Turtle-Dokument gezeigt.

```
1  @prefix potter: <http://143.93.114.104/rest/samian/potters/> .
2  @prefix samian: <http://143.93.114.104/rest/samian/> .
3  potter:Balbinus_i samian:Werkstatt <http://www.potterwerkstatt.de/001> .
4  potter:Balbinus_i samian:Geschlecht <http://www.geschlecht.info/male> .
5  potter:Balbinus_i samian:Name "Balbinus i" .
6  potter:Balbinus_i samian:Alter 21 .
7  potter:Balbinus_ii samian:Werkstatt <http://www.potterwerkstatt.de/001> .
8  potter:Balbinus_ii samian:Geschlecht <http://www.geschlecht.info/male> .
9  potter:Balbinus_ii samian:Name "Balbinus ii" .
10 potter:Balbinus_ii samian:Alter 30 .
11 potter:Balbinus_iii samian:Werkstatt <http://www.potterwerkstatt.de/002> .
12 potter:Balbinus_iii samian:Geschlecht <http://www.geschlecht.info/female> .
13 potter:Balbinus_iii samian:Name "Balbinus iii" .
14 potter:Balbinus_iii samian:Alter 55.
```

Das Dokument zeigt drei verschiedene Töpfer, welche jeweils einen Namen besitzen und von denen das Geschlecht und ihr Alter bekannt sind. Zwei dieser Töpfer arbeiten in der Werkstatt 001, eine Töpferin in Werkstatt 002. Die Anfrage würde folgendes Ergebnis liefern:

name	**geschlecht**
“Balbinus ii”	http://www.geschlecht.info/male
“Balbinus i”	http://www.geschlecht.info/male

Die Verwendung des Schlüsselwortes SELECT erzeugt eine Tabelle, welche das zu erwartende Ergebnis zeigt. Wäre im SELECT-Statement eine Wilcard vergeben, würde das Ergebnis wie folgt aussehen:

potter	**geschlecht**	**name**
http://143.93.114.104/rest/samian/potters/Balbinus_ii	“Balbinus ii”	http://www.geschlecht.info/male
http://143.93.114.104/rest/samian/potters/Balbinus_ii	“Balbinus i”	http://www.geschlecht.info/male

Mit den bereits vorgestellten Mitteln können grundlegende Muster in RDF-Dokumenten gesucht werden. Oftmals reicht dies für praktische Anwendungen jedoch nicht aus. Eine Suche, z.B. nach einem Altersbereich, nicht nach einem bestimmten Alter stellt ein Problem, der bislang vorstellten Methoden dar. Bislang besteht keine Möglichkeit Literale (Text oder Zahlen, keine URI) zu suchen und nach ihnen zu filtern. Diese Anfrage ist in SPARQL durch so genannte Filterbedingungen möglich. Durch zusätzliche Bedingungen in der Anfrage kann die Ergebnismenge eingeschränkt werden. Filter können mit verschiedenen Operatoren angereichert werden: Vergleichsoperatoren (=, >, <, >=, <=, !=), spezielle Operatoren (z.B. isLiteral(A) oder isURI(A)), boolsche Operatoren (&&, ||, !) oder arithmetische Operatoren (+, -, *, /) [HITZLER ET AL., 2008]. So kann in der zuvor erwähnten Turtle Datei nach Töpfern und deren Alter gesucht werden, die zwischen 25 und 60 Jahren alt sind:

```
1 PREFIX potter: <http://143.93.114.104/rest/samian/potters/>
2 PREFIX samian: <http://143.93.114.104/rest/samian/>
3 SELECT ?potter ?alter
4 WHERE {
5   ?potter samian:Alter ?alter .
6   FILTER (?alter >=25 && ?alter <=60)
7 }
```

Zeile 3 definiert die Ausgabevariablen: URI des Töpfers und das Alter jener. In Zeile 5 wird durch eine Abfrage sichergestellt, dass einem Töpfer ein Alter zugewiesen ist. Die Filterbedingung ist in Zeile 6 eingeführt. Hierbei kommt eine Kombination zwischen Vergleichoperatoren und boolschen Operatoren zum Einsatz, die einen Bereich definieren. Die Anfrage ergibt folgendes Ergebnis:

potter	**alter**
http://143.93.114.104/rest/samian/potters/Balbinus_iii	55^^http://www.w3.org/2001/XMLSchema#integer
http://143.93.114.104/rest/samian/potters/Balbinus_ii	30^^http://www.w3.org/2001/XMLSchema#integer

SPARQL-Anfragen können weitere Einschränkungen, ähnlich zu SQL Abfragen, enthalten, z.B. GROUP BY, ORDER BY und LIMIT. Größere Übersichten sind im Internet[104] verfügbar. Im

[104] siehe auch http://rdf.myexperiment.org/howtosparql (abgerufen 03.12.2013)

Sprachgebrauch ist oft von einem SPARQL Endpoint zu hören. Dieser stellt im einfachsten Fall einen URI dar, die SPARQL konforme Anfragen verarbeiten kann und passende Ergebnisse zurückliefert[105]. Weitergehende Informationen können der offiziellen Spezifikation[106] oder auch HITZLER ET AL., (2008), Seite 202 bis 241, entnommen werden.

SPARQL Anfragen werden in dieser Arbeit u. A. zur Abfrage der Beziehungen zwischen Töpfern, Findspots und Keramikfragmenten, wie auch der Verlinkungen zu Pleiades Places genutzt.

2.5.5 Triple Stores

Unter einem RDF- bzw. Triple Store versteht man eine relationale Datenbank, die speziell zur effizienten Speicherung von RDF-Tripeln, zum Abfragen und allgemeiner Interaktion mit Tripeln ausgelegt ist. Nach Außen repräsentiert der Triplestore lediglich eine einzelne flache Tabelle mit drei Spalten für Subjekte, Prädikate und Objekte (vgl. Abbildung 2-22). Die interne Struktur des Triple Stores jedoch kann als wesentlich komplexer angesehen werden. Eine große Anzahl an Tripeln erfordert beispielsweise eine Indexierung zu einer effizienten Befragung des Stores [HAFT, 2013].

S	P	O
anno:Patricius_i-Chesterholm-Vindolanda	oa:hasBody	findspot:Chesterholm-Vindolanda
anno:Patricius_i-Chesterholm-Vindolanda	oa:hasTarget	potter:Patricius_i
anno:Patricius_i-Chesterholm-Vindolanda	dcterms:description	"Link made by Heinz/Mees/Thiery. Database by RGZM. Potter has worked in Findspot or exported to it."

Abbildung 2-22: Triplestore (Subjekt, Prädikat, Objekt)

Auf Grund der Aufgabenstellung können Kriterien an den zu nutzenden Triple Store gestellt werden, die nachfolgend aufgelistet sind (vgl. auch HAFT (2013), da in dem behandelten IBR Projekt ähnliche Anforderungen gestellt werden):

- Der Store sollte unter Open Source Lizenzen stehen als freie Software erhältlich sein. Finanzielle Mittel öffentlicher Träger sind begrenzt, daher sind kostengünstige Lösungen zu bevorzugen. Open Source Software wird im Allgemeinen als sicher angesehen, eine Bindung an einen Anbieter findet nicht statt. Offene Quellcodes ermöglichen einfache Modifizierungen und dienen der Nachvollziehbarkeit der Prozesse.
- Der Store sollte in einer verbreiteten Programmiersprache erstellt worden sein. Somit ist die Möglichkeit der Anbindung von Plug-Ins, etc. möglich.
- Der Store solle SPARQL Anfragen annehmen können, damit der Datenbestand gezielt befragt werden kann.
- In dieser Arbeit wird eine ReST-Schnittstelle erzeugt. Eine ReST-Funktionalität des Stores ist deshalb ebenfalls wünschenswert.
- Ein Produkt mit einer aktiven Community, welche sich um eine Weiterentwicklung sorgt sollte bevorzugt werden.

---

[105] siehe auch (abgerufen 03.12.2013) http://answers.semanticweb.com/questions/3629/what-is-sparql-endpoint

[106] vgl. http://www.w3.org/TR/sparql11-overview (abgerufen 03.12.2013)

- Der Installationsaufwand sollte gering sein
- Eine komfortable Interaktion mit dem Store sollte mit einer benutzerfreundlichen Administrationsoberfläche möglich sein.

Haft vergleicht in seiner Bachelorarbeit drei verschiedene Stores in verschiedenen Programmiersprachen: Semsol ARC2 (PHP), Sesame (JAVA) und Redland (C). Eine Gegenüberstellung dieser wird in Abbildung 2-23 aufgezeigt:

	Semsol ARC2	**Sesame**	**Redland**
Freie / Open Source Software	Ja	Ja	Ja
Geschrieben in	PHP	Java	C
Erweiterbar / Plugin fähig	Ja	Ja	Ja
Turtle-Syntax Support	Ja	Ja	Ja
REST-Schnittstelle	Nein	Ja	Nein
SPARQL-Support	Ja	Ja	Ja
Konfigurations- und/oder Administrationsoberfläche (optional)	Nein	Ja	Nein
Aktive Weiterentwicklung	Nein	Ja	Ja
Installationsaufwand	gering	mittel	Sehr hoch

Abbildung 2-23: Triplestorevergleich [HAFT, 2013]

Der Vergleich zeigt, dass Sesame als einziger Triple Store die aufgestellten Kriterien erfüllt. Zudem ist er in JAVA entwickelt. Dies stellt einen weiteren positiven Aspekt dar, da alle Anwendungen des GeInArFa Projekts mittels JAVA realisiert werden.

OpenRDF Sesame[107] ist ein Open Source System bzw. Framework, welches in JAVA entwickelt wird, mit dem Ziel Daten in RDF zu verwalten und abzufragen [OPENRDF, 2013A]. Sesame zeichnet sich durch seine Pluginfähigkeit, Turtle- und SPARQL-Support, sowie über die integrierte ReST-Schnittstelle aus [OPENRDF, 2013B]. Die Benutzung des Stores in vielen Bereichen kann als weiterer positiver Punkt aufgenommen werden (OPENRDF, 2013C). Die Administrationsoberfläche Open Sesame Workbench[108] ist intuitiv und bietet eine Reihe von Möglichkeiten, wie das Hinzufügen von Daten und Abfragen mittels SPARQL. Der frei zugängliche Quellcode[109] kann zudem aus dem Github Repository bezogen werden. Sesame kann als WAR-File[110] auf dem in dieser Arbeit verwendeten Apache Tomcat (vgl. Kapitel 4) installiert werden. Die Arbeit mit dem Workbench[111]

[107] vgl. http://www.openrdf.org (abgerufen 03.12.2013)

[108] siehe auch https://wiki.csiro.au/display/~yu021/SESAME+Repository+Management+using+Openrdf-Workbench (abgerufen 03.12.2013)

[109] vgl. https://bitbucket.org/openrdf/sesame/src (abgerufen 03.12.2013)

[110] siehe http://sourceforge.net/projects/sesame/files/Sesame%202/ (abgerufen 03.12.2013)

[111] siehe auch https://wiki.csiro.au/display/~yu021/SESAME+Repository+Management+using+Openrdf-Workbench (abgerufen 03.12.2013)

kann u. A. in Haft (2013) und Kapitel 6.3 nachvollzogen werden. Zu weitergehenden Informationen sei an dieser Stelle auf die Dokumentation[112] verwiesen.

Der Sesame Triple Store wird in dieser Arbeit zur Ablage der erstellten Triple genutzt. Des Weiteren werden über Ihn Graphen-Abfragen zu Beziehungen zwischen Töpfern, Findspots und Keramikfragmenten, sowie zu Verlinkungen zum Pleiades Projekt, getätigt.

2.5.6 Web Ontology Language (OWL)

RDF (vgl. Kapitel 2.5.3) bzw. RDFs (vgl. Kapitel 3.2.1) reichen zu einer Modellierung einfacher Ontologien und einer Ableitung von implizitem Wissen aus. Beide Schemata besitzen jedoch nur beschränkte Ausdrucksmittel und können komplexe Zusammenhänge nicht modellieren. Die Darstellung komplexen Wissens wird mit Hilfe von ausdrucksstarken Repräsentationssprachen, die auf formaler Logik basieren, gelöst. Eine dieser Sprachen ist die Ontologiesprache OWL[113] [HITZLER ET AL., 2008]. OWL wird in dieser Arbeit nicht verwendet. Das folgende Kapitel ist daher als Ergänzung zu Techniken wie RDF und RDFs zu sehen.

OWL steht für das Akronym Web Ontology Language. Der Ursprung des Buchstabentausches (WOL zu OWL) gibt Platz zu Spekulationen. Offensichtlich jedoch, geht dieser auf eine E-Mail[114] des Professors Tim Finin[115] der University of Maryland zurück. Gründe hierfür sind die einfache Aussprache (engl. für Eule), damit die hervorragende Eignung zur Erstellung eines Logos, Eulen werden mit Weisheit assoziiert und es eine interessante Hintergrundgeschichte. In einem Projekt von William A. Martin in den 70er Jahren der vorangegangen Jahrhunderts des MIT mit dem Namen One World Language wurde bereits ein Versuch der Entwicklung einer universellen Sprache zur Wissensrepräsentation gestartet. Die Behauptung das Akronym sei eine Referenz auf eine Eule, welche in Alan Alexander Milnes Buch Winnie the Pooh auftritt und ihren Namen fälschlicherweise WOL anstatt OWL schreibt, stellt jedoch eine weitere interessante Behauptung dar[116] [HITZLER ET AL., 2008].

OWL wurde 2004 vom W3C als Ontologiesprache standardisiert und kann heute auf eine hohe Akzeptanz in vielen Anwendungsbereichen blicken. Das Gleichgewicht zwischen Ausdrucksstärke der Sprache und einer effizienten Möglichkeit der Schlussfolgerung spielte bei der Schaffung der Web Ontology Language eine große Rolle. Sprachkonstrukte, die komplexe Ausdrucksmöglichkeiten für implizites Wissen zur Verfügung stellen, führen gleichzeitig zu hohen Komplexitäten und schlechten Skalierbarkeitseigenschaften. Auf Grund verschiedener Anforderungen in der Praxis gibt es verschiedene Teilsprachen von OWL: OWL Full, OWL DL und OWL Lite [HITZLER ET AL., 2008].

---

[112] vgl. http://openrdf.callimachus.net/sesame/2.7/docs/system.docbook?view (abgerufen 03.12.2013)

[113] vgl. http://www.w3.org/TR/owl-features (abgerufen 03.12.2013)

[114] vgl. http://lists.w3.org/Archives/Public/www-webont-wg/2001Dec/0169.html (abgerufen 03.12.2013)

[115] siehe auch http://www.csee.umbc.edu/~finin/ (abgerufen 03.12.2013)

[116] siehe auch http://www.w3.org/2003/08/owlfaq (abgerufen 03.12.2013)

OWL-Ontologien werden in OWL Dokumenten beschrieben. Zwei verschiedene Syntaxen, OWL-RDF-Syntax (basierend auf RDF als Datenaustausch) und OWL-Syntax (abstrakt, für Menschen einfacher zu lesen) vereinfachen den Umgang mit diesen Ontologien. Eine OWL-Ontologie besteht aus Klassen und Propertys, vgl. RDF(s). Im Vergleich hierzu können diese jedoch in komplexe Beziehungen (mit Hilfe von Konstruktoren) gesetzt werden. Neben Klassen und Propertys werden auch Individuen (RDF-Instanzen) verwendet, OWL Propertys werden oft auch als Rollen bezeichnet [HITZLER ET AL., 2008].

Das nachstehende Beispiel folgt diversen Ausführungen in HITZLER ET AL. (2008) und soll einen ersten Eindruck der Ontologiesprache und deren Konstrukte vermitteln. Die Klasse Potter kann mit nachfolgender OWL-RDF-Syntax erzeugt werden.

```
1 <owl:Class rdf:about="Potter" />
```

Ähnlich zu RDF können Individuen als Instanzen von Klassen deklariert werden (Balbinus ist ein konkreter Töpfer):

```
1 <owl:Potter rdf:about="Balbinus" />
```

Einfache Klassenbeziehungen (Hierarchien) mittels rdfs:subClassOf sind ebenfalls möglich:

```
1 <owl:Class rdf:about:"Potter">
2   <rdfs:subClassOf rdf:resource="antikePerson" />
3 </owl:Class>
```

Zwei Arten von Rollen können Individuen mit Individuen (abstrakt) oder Individuen mit Datenwerten (konkret) verbinden. Beides kann als Unterklasse von rdf:Property angesehen werden. Des Weiteren können Bereiche, auf die sich Rollen beziehen, mittels rdfs:domian und rdfs:range (vgl. RDFs) explizit deklariert werden:

```
1 <owl:ObjectProperty rdf:about="Zugehörigkeit">
2   <rdfs:domain rdf:resource="Potter" />
3   <rdfs:range rdf:resource="Organisation" />
4 </owl:ObjectProperty>
5 <owl:ObjectProperty rdf:about="Name">
6   <rdfs:domain rdf:resource="Potter" />
7   <rdfs:range rdf:resource="&xsd;string" />
8 </owl:ObjectProperty>
```

Die Rolle Zugehörigkeit ist abstrakt und soll die Zugehörigkeit eines Töpfers zu einer bestimmten Organisation ausdrücken. Die Rolle Name hingegen weist einen konkreten Namen als String zu:

```
1 <Potter rdf:about:"Balbinus" />
2   <Zugehörigkeit rdf:resource="OrganisationX" />
3   <Name rdf:datatype="&xsd;string"/>Balbinus</Name>
4 </Potter>
```

Weitere Informationen und Modellierungsmöglichkeiten sind HITZLER ET AL. (2008) zu entnehmen.

3 Interoperabilität und Semantik in archäologischen Fachdaten

Datenbanken archäologischer und anderer geisteswissenschaftlicher Fachdaten liegen im World Wide Web[117] teilweise in diversen Datenbanken vor. Eine Zusammenarbeit in Forschung und Entwicklung ist jedoch nur über definierte Schnittstellen möglich. Diese Interoperabilität wird durch Standards und Quasi-Standards, hier die allgemeine Nutzung von Repräsentationen in vielen LOD-Projekten welche keiner Festlegung eines Gremiums bedürfen, gewährleistet. Die Repräsentation der eigenen Daten in (quasi-) standardkonformen Formaten bedingt hierdurch nicht nur ein mögliches Data-Sharing, sondern insbesondere die Nutzung der Vorteile im eigenen Projekt. Eine semantische Auszeichnung der Daten bewirkt zudem ein größeres und besseres Verständnis der Daten in den eigenen und anderen Forschungsgruppen. Die folgenden Kapitel geben einen Überblick der verwendeten Standards der Interoperabilität und Semantik im Bereich der archäologischen Fachdaten, abgestimmt auf die Nutzung in dieser Masterarbeit. Sie zeigen das Ergebnis von Internetrecherchen (bis Mitte November 2013) und erheben keinen Anspruch auf Vollständigkeit.

3.1 XML- und JSON-Auszeichnungen

Geisteswissenschaftliche Repositorien stellen ihre Daten oft in einem XML[118]-Schema dar. Dies bietet den Vorteil, dass XML als W3C[119]-Standard eine breite Akzeptanz in diversen Projekten der Informationstechnik zum Austausch und der Repräsentation von Daten genießt.

Eine Auszeichnung archäologischer Fachdaten kann mit dem speziell auf diesen Fall zugeschnittenen Standard ArchaeoML erfolgen. Zum Zeitpunkt der Recherche jedoch ist keine Dokumentation dieses Formats mehr verfügbar. Die URI des Standards[120] (hier am Beispiel einer Open Context Ressource[121]) führt zu einem toten Link. In diesem Fall ist von einer Verwendung abzuraten (siehe auch Diskussionen in Internetforen[122]).

Bibliotheken und Bibliotheksstandards nehmen in den Geisteswissenschaften eine führende Rolle ein. Sie beherbergen eine große Masse an physischen Daten (wie Bücher, Bilder, archäologische

---

[117] siehe auch http://users.minet.uni-jena.de/~sack/WWWBuch (abgerufen 03.12.2013)

[118] vgl. http://www.w3.org/XML (abgerufen 03.12.2013)

[119] vgl. http://www.w3.org (abgerufen 03.12.2013)

[120] vgl. http://ochre.lib.uchicago.edu/schema/SpatialUnit/SpatialUnit.xsd (abgerufen 03.12.2013)

[121] vgl. http://opencontext.org/subjects/67BDF2BC-E33C-4D99-9947-5D15D78A45E6 (abgeruf. 03.12.2013)

[122] siehe auch https://groups.google.com/forum/#!forum/antiquist (abgerufen 03.12.2013)

Objekte) und sind somit auch der Verwaltung dieser, insbesondere mit Metadaten, verpflichtet. Aufgrund dieser Tatsache können XML-Auszeichnungen der Bibliotheken als Auszeichnungsstandard der eigenen Daten zur Interoperabilität beitragen, da jener Standard von vielen anderen Forschern und Forschungsgruppen verwendet und verstanden wird und somit Synergien erzeugen kann.

MADS[123] (Metadata Authority Description Standard), welcher von der Library of Congess[124] (Washington DC) betrieben wird, ist eines dieser Bibliotheksstandards (siehe auch MARC 21). Ursprünglich ist MADS als Schema für ein authority element set von Bibliotheken gedacht, das Metadaten über "agents", "events" und "terms" speichert. Des Weiteren ist ebenfalls die Möglichkeit der Transformationen jener Standards in RDF möglich [MADS, 2013].

Ein zweites Standbein zur Auszeichnung von archäologischen Fachdaten können neben Bibliotheken auch Organisationen zur Erforschung und Erhaltung von Kulturerbe sein. In Großbritannien übernimmt dies English Heritage[125]. MIDAS-Heritage ist ein britischer Standard der English Heritage, u. A. zur Aufzeichnung von Informationen über Gebäude, archäologischen Fundstätten, und Artefakten. Er wird z.B. von nationalen Regierungsorganisationen in Großbritannien, Organisationen des Kulturerbes in der Forschung und von Unternehmen zum Austausch von Wissen und zur Langzeitarchivierung eingesetzt [ENGLISHHERITAGE, 2013]. MIDAS XML ist ein W3C[126]-XML Schema, das ein gemeinsames Format für die Speicherung, Verarbeitung und den Austausch von historischen Informationen bereitstellt. Es deckt damit alle gesammelten Informationen des MIDAS-Heritage[127] Standard ab [HERITAGESTANDARDS, 2013]. MIDAS XML ist ein Ergebnis des FISH[128] (Forum on Information Standards in Heritage) und liegt in der aktuellsten Version 2.0 [HERITAGESTANDARDS, 2013] in diversen Schemata[129] und einer Dokumentation[130] vor und bietet u. A. die Möglichkeit der Auszeichnung von Objekten, Monumenten, Events und Personen. Die Einbindung kontrollierter Vokabulare ist ebenfalls möglich [LEE, 2013]. Details zu diesen Standards und Beispiele sind Kapitel 6.1.5 zu entnehmen.

Liegen Daten nicht als Objekte, bzw. historische Personen, sondern als Texte vor, kann die Text Encoding Initiative[131] (TEI), welches ein Konsortium zur Entwicklung und Wartung eines Standards zur Repräsentation von Texten in digitaler Form ist, eine Lösung darstellen. Diese Richtlinien zur Kodierung maschinenlesbarer Texte in den Geistes-, Sozial- und Sprachwissenschaften werden

---

[123] vgl. http://www.loc.gov/standards (abgerufen 03.12.2013)

[124] vgl. http://www.loc.gov/index.html (abgerufen 03.12.2013)

[125] vgl. http://www.english-heritage.org.uk (abgerufen 03.12.2013)

[126] vgl. http://www.w3.org (abgerufen 03.12.2013)

[127] vgl. http://www.english-heritage.org.uk/publications/midas-heritage/midas-heritage-2012-v1_1.pdf

[128] vgl. http://www.heritage-standards.org.uk (abgerufen 03.12.2013)

[129] vgl. http://www.heritage-standards.org.uk/midas/schema/2.0 (abgerufen 03.12.2013)

[130] vgl. http://www.heritage-standards.org.uk/midas/docs (abgerufen 03.12.2013)

[131] vgl. http://www.tei-c.org/index.xml (abgerufen 03.12.2013)

weitestgehend von Bibliotheken, Museen und Verlagen verwendet[132] [TEI, 2013]. Eine dieser Initiativen ist EpiDoc[133] (EpiDoc: Epigraphic Documents in TEI XML). EpiDoc spezialisiert sich auf die Veröffentlichung von digitalen Ausgaben alter Inschriften, deren Geschichte und Mentalität der Objekte [EPIDOC, 2013]. Beispiele[134] zur Anwendung dieses Standards können in der DIO[135] (Deutsche Inschriften Online) ReST-Schnittstelle abgerufen werden.

Im Vergleich zur Bereitstellung von Informationen der Geisteswissenschaften (hier insbesondere archäologische Objekte) ist eine räumliche Darstellung von Geometrien in standardisierten Formaten einfacher zu realisieren. Das Open Geospatial Consortium[136] stellt diverse Formate und Dienste (vgl. Kapitel 2) im Bereich der Geoinformation zur Verfügung. Eines dieser Formate ist GML[137] (Geography Markup Language). Die Repräsentation von Geometrien und deren Eigenschaften (Simple Feature) ist somit standardisiert (z.B. mit Geoserver und Geotools) erzeugbar und mit den meisten geometrieverarbeitenden Programmen bearbeitbar.

GML ist eine XML-Auszeichnungssprache zur Beschreibung und Austausch geografischer Objekte und dient sowohl als Modellierungssprache geografischer Systeme, wie auch als offenes Austauschformat für räumliche Transaktionen über das Internet. Insbesondere das generische Objektmodell, welches konventionelle Vektordaten, einfache Objekte, wie auch multidimensionale Rasterdaten und Elemente von Sensordaten enthält und die Fähigkeit beinhaltet, alle Arten von räumlichen Informationen zu integrieren, ist ein Schlüssel zur Nützlichkeit von GML. Die Geography Markup Language beinhaltet eine Vielfalt von Primitiven wie Objekte (Features), Geometrien, Koordinatensysteme, Topologien, Zeit, dynamische Objekte, Rasterdaten, Maßstäbe, Richtungen, Beobachtungen und Regeln zur Kartengestaltung. Als aktuelle Version ist 3.2.1, an einer Version 3.2.2 und 4 wird jedoch gearbeitet [OSGEO, 2013]. GML liegen die Geometrien der Simple Feature Specification[138] zugrunde und kann mit Xpointer[139] bzw. Xlink[140] angereichert werden und somit Verlinkungen erzeugen [FELICETTI ET AL., 2007].

Im Vergleich zu XML sind Darstellungen in JSON[141] (JavaScript Object Notation) nicht in diesem Maße standardisiert. Für die zuvor vorgestellten XML Auszeichnungen sind keine adäquaten standardisierten JSON Modellierungen (bzw. Transformationen) verfügbar. JSON ist ein Datenaustauschformat, das für Menschen und Maschinen leicht zu lesen und zu erzeugen ist und basiert auf

---

[132] siehe auch Liste der Projekte: http://www.tei-c.org/Activities/Projects (abgerufen 03.12.2013)

[133] vgl. http://sourceforge.net/p/epidoc/wiki/Home (abgerufen 03.12.2013)

[134] ein Beispiel: http://www.inschriften.net/rest/di019/articles/di019-0001 (abgerufen 03.12.2013)

[135] vgl. http://www.inschriften.net (abgerufen 03.12.2013)

[136] vgl. http://www.opengeospatial.org (abgerufen 03.12.2013)

[137] vgl. http://www.opengeospatial.org/standards/gml (abgerufen 03.12.2013)

[138] vgl. http://www.opengeospatial.org/standards/sfa (abgerufen 03.12.2013)

[139] vgl. http://www.w3.org/TR/xptr (abgerufen 03.12.2013)

[140] vgl. http://www.w3.org/TR/xlink (abgerufen 03.12.2013)

[141] vgl. http://www.json.org (abgerufen 03.12.2013)

einer Teilmenge von JavaScript. Es ist (programmier-) sprachenunabhängig, nutzt jedoch Konventionen aus C-Programmiersprachen (z.B. C++, C#, Java, Python) und kann somit als Austauschsprache genutzt werden [JSON, 2013]. Geographische Objekte, inkl. deren Attribute und Eigenschaften, können analog zu GML als GeoJSON[142] dargestellt werden. GeoJSON ist ein allgemein anerkanntes Format zur Kodierung von geografischen Datenstrukturen [GEOJSONWIKI, 2013]. Die so erzeugten GeoJSON Objekte können sowohl eine Geometrie, ein Feature (Geometrie + Eigenschaften), wie auch eine Gruppe von Features mit den Typen Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon und GeometryCollection (vgl. Simple Feature Specification) beschreiben [BUTLER ET AL., 2008]. Die so erzeugten Geometrien können durch die optionale Verwendung (Standard ist WGS84) eines Coordinate Reference Systems (CRS) erweitert werden und somit die Lage in verschiedenen geodätischen Referenzsystemen abbilden [BUTLER ET AL., 2008].

Die Tate Gallery[143] zeigt eine Möglichkeit der JSON-Modellierung[144] von Künstlerattributen und deren Werke[145] auf [TATE, 2013]. Das JSON-Objekt[146] gibt hier einen besseren (hierarchischen) Überblick als einfache CSV Dateien[147]. Eine Standardisierung ist jedoch nicht zu erkennen.

3.2 Linked Data

Semantik und Interoperabilität in archäologischen Fachdaten lässt sich besonders mit Hilfe des Linked Data Ansatzes aufzeigen. Linked Data, als Teil des Semantic Web, bieten per se offene und semantisch ausgezeichnete Daten an (siehe auch Prädikatenlogik[148]). Dies umgeht somit geschickt das Problem der Closed World Assumption[149] in der Informatik.

3.2.1 Technologien

Zur Lösung der Bereitstellung von interoperablen Linked Data können insbesondere die Technologien RDF/RDFs/RDFa, SKOS und Open Annotation beitragen, welche auch in den nachfolgend beschriebenen Projekten Anwendung finden.

---

[142] vgl. http://geojson.org (abgerufen 03.12.2013)

[143] vgl. http://www.tate.org.uk (abgerufen 03.12.2013)

[144] vgl. http://www.tate.org.uk/about/our-work/digital/collection-data (abgerufen 03.12.2013)

[145] vgl. https://github.com/tategallery/collection (abgerufen 03.12.2013)

[146] Beispiel: https://github.com/tategallery/collection/blob/master/artworks/a/000/a00001-1035.json (abgerufen 03.12.2013)

[147] Beispiel: https://github.com/tategallery/collection/blob/master/artwork_data.csv (abgerufen 03.12.2013)

[148] siehe auch: http://www.informatik.uni-hamburg.de/WSV/teaching/vorlesungen/FGI1SoSe07/PL.pdf (abgerufen 03.12.2013)

[149] vgl. http://www.sciencedirect.com/science/article/pii/S0022000005800042 (abgerufen 03.12.2013)

Die Technik des Ressource Description Framework (RDF), inklusive formalisierter Aussagen über Ressourcen, deren Darstellung als Triple und die Abbildung in einem Graphen werden bereits in Kapitel 2.5.3 erläutert.

RDF-Schema (RDFs[150]) ist ein spezielles RDF-Vokabular. Im Vergleich zu thematischen Vokabularien wie z.B. FOAF stellt es keine neuen Termini für spezielle Anwendungsdomänen zur Verfügung, vielmehr universelle Ausdrucksmittel, die ermöglichen, innerhalb eines RDFs-Dokuments Aussagen über die semantischen Beziehungen der Termini eines beliebigen nutzerdefinierten Vokabulars zu machen. Somit lässt sich RDFs als Wissensrepräsentations- oder Ontologiesprache beschreiben, mit der es möglich ist, in der Domäne bereits vorhandene Abhängigkeiten zu darzustellen, vgl. HITZLER ET AL. (2008). RDFs stellt Ressourcen (ressources), Klassen (classes), Eigenschaften (properties) und Hierarchien (rdfs:subClassOf und rdfs:subPropertyOf) für RDF zur Verfügung. Definitions- und Wertebereiche (Contraints; rdfs:domain und rdfs:range) für klassifizierte Ressourcen, wie auch Annotationsmöglichkeiten (rdfs:label, rdfs:comment rdfs:seeAlso) sind ebenfalls Bestandteile von RDF-Schema. All jene Features von RDFs bieten die Möglichkeit, von einer reinen RDF-Wissensrepräsentation zu einer semantischen Beschreibung überzugehen [HAFT, 2013]. Mit RDF-Schema kann somit die Erzeugung einer eigenen Ontologie mit Klassen- und Klassenhierarchie, sowie Property- und Propertyhierarchien inkl. Constraints unterstützt werden. Ein kleines Beispiel (Klasse Töpfer mit der Eigenschaft Geschlecht) zur Verdeutlichung des Verfahrens ist nachstehend in Turtle Syntax gegeben (analog zu einem Beispiel aus HAFT, 2013). Weitere Sprachmittel des RDF-Schema sind der offiziellen Webressource[151] zu entnehmen.

```
#Prefix
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix myclass: <http://meineklasse.de/> .
@prefix myproperty: <http://meinepraedikat.de/>.
@prexix samian: <http://143.93.114.104/rest/samian/potters/> .

#Klassen und abgeleitete Klassen inkl. Annotationen
myclass:Person rdf:type rdfs:class .
myclass:Person rdfs:subClassOf rdfs:Ressource .
myclass:Potter rdf:type rdfs:class .
myclass:Potter rdfs:subClassOf myclass:Person .
myclass:Potter rdfs:label “Potter”@en .
```

[150] vgl. http://www.w3.org/TR/rdf-schema (abgerufen 03.12.2013)

[151] vgl. http://www.w3.org/2000/01/rdf-schema# (abgerufen 03.12.2013)

```
myclass:Potter rdfs:label “Toepfer”@de .
myclass:Potter rdfs:comment “Ein Töpfer ist eine Person, welcher ein Keramikwerkstückt hergestellt hat”@de .
samian:Balbinus rdf:type myclass:Potter .

# Propertys und abgeleitete Eigenschaften
myproperty:Geschlecht rdf:type rdf:Property .
myproperty:male rdfs:subPropertyOf myproperty:Geschlecht .
myProperty:male rdfs:domain myclass:Person .
myproperty:male rdfs:range myclass:Person .
```

Das Beispiel zeigt die Definition und Typisierung der Ressource myclass:Person als Klasse. Die Hierarchie innerhalb dieser Klasse (Potter ist eine Subklasse von Person und der vordefinierten Klasse Ressource) wird durch rdfs:subClassOf ausgedrückt. Klassenstrukturen in RDF sind transitiv, so dass das konkrete semantische Objekt samian:Balbinus dem Typ myclass:Potter, myclass:Person und rdfs:Ressouce entspricht. Entsprechendes Labeling (hier in Englisch und deutsch) und Kommentieren der Ressourcen (hier in Deutsch) erlaubt eine ausführliche Beschreibung. Einem Töpfer können mehrere Eigenschaften zugewiesen werden, u. A. das Geschlecht. Dieser Sachverhalt kann mit Hilfe von RDFS-Sprachmitteln (hier rdf:Property und rdfs:subPropertyOf) formalisiert werden. Constraints legen in diesem Fall fest, dass die Eigenschaft Geschlecht (und durch die Transitivität auch dessen sub-Eigenschaft male) der Klasse Person (oder auch der Subklasse Potter) zugeordnet ist und als Wert ebenfalls nur ein semantisches Objekt der Klasse Person zulässig ist. Letztendlich kann mit Hilfe dieser Ontologie dem semantischen Objekt samian:Balbinus das Geschlecht myproperty:male zugeordnet werden.

Darüber hinaus gibt es die Möglichkeit Linked Data in HTML mittels RDFa[152] zu integrieren. RDFa ist eine HTML5 Erweiterung zur Markierung von Personen, Menschen, Events und Bewertungen. Suchmaschinen nutzen diese Makups zum Beispiel zur Generierung besserer Suchergebnisse. RDFa kann jedoch auch in anderen XML basierenden Dokumenten verwendet werden [RDFA, 2013]. Die ausgezeichneten Ressourcen werden of mit schema.org[153] typisiert [SPORNY ET AL., 2012], da dies besonders von gängigen Suchmaschinen identifiziert wird [SCHEMA, 2013]. Nachfolgend sein ein Beispiel[154] zur Verdeutlichung gegeben.

[152] vgl. http://www.w3.org/TR/xhtml-rdfa-primer (abgerufen 03.12.2013)

[153] vgl. http://schema.org (abgerufen 03.12.2013)

[154] anlehnend an http://www.w3.org/TR/rdfa-lite (abgerufen 03.12.2013)

```
<p vocab="http://schema.org/" prefix="ov: http://open.vocab.org/terms/"
resource="Balbinus" typeof="Person">
   My name is
   <span property="name">Balbinus</span>
   and you can give me a ring via
   <span property="telephone">1-800-555-0199</span>.
   <img property="image"
src="http://www.myartprints.co.uk/kunst/roman/portrait_balbinus_d238_hi.jpg"
/>
</p>
```

Eine weitere grundlegende Technologie zur Erstellung einer eigenen Ontologie und Erzeugung von Austauschbarkeit unter diesen Ontologien ist SKOS. SKOS (Simple Knowledge Organization System) ist ein W3C Standard, welcher auf anderen Semantic Web-Standards wie RDF und OWL basiert. Es stellt insbesondere eine Basisstruktur für den Inhalt von Schemata wie Thesausi, Klassifikation und kontrollierten Vokabularien dar. Das fundamentale Element des SKOS Vokabulars ist das Concept. Concepts können z.B. Objekte oder Events sein, es ist jedoch nicht direkt eine hierarchische Struktur ableitbar. SKOS ermöglicht die Veröffentlichung von Konzepten im World Wide Web und die Verlinkungen[155] zu anderen Concept Schemata [ISAAC ET AL., 2009]. Concepts können via URIs identifiziert werden. Ein Beispiel in Turtle Syntax demonstriert die Verwendung von SKOS (ohne Angabe der Prefixes):

```
ex:Potter rdf:type skos:Concept;
    skos:prefLabel "Potter"@en;
    skos:prefLabel "Töpfer"@de;
    skos:altLabel "Potter"@en;
    skos:exactMatch ex2:Potter2.
```

Das Beispiel zeigt einen Teil des Concepts Potter, welcher in verschiedenen Sprachen eine Beschreibung enthält und einem anderem im WWW verfügbaren Konzept Potter2 exakt entspricht. Eine hierarchische Abbildung und Beschreibung von Attributen (Prädikaten) eines Concepts wäre nur in Kombination mit RDFs möglich (z.B. ex:potter rdf:type rdfs:class, etc.).

Eine dritte wichtige Technologie ist Open Annotation. Eine Annotation ist sowohl für Geisteswissenschaftler, wie auch den Naturwissenschaftler bzw. Ingenieure ein weit verbreitetes Element der wissenschaftlichen Praxis. Es dient vor allem zur Einordnung und Generierung existierendem Wissens, sowie der gemeinsamen Nutzung neuen Wissens. Eine Annotation kann in sowohl ein Kommentar, als auch eine Methode zur Klassifizierung oder Verlinkung sein. Die Open Annotation Collaboration[156] (OAC) zeigt einen Ansatz in der Generierung einer interoperablen Spezifikation

[155] vgl. http://www.w3.org/TR/skos-primer/#secrel (abgerufen 03.12.2013)

[156] vgl. http://www.openannotation.org (abgerufen 03.12.2013)

mit dem Ziel Annotationen über das Web plattformunabhängig auszutauschen. Zehn tiefergreifende Leitprinzipien sind den Open Annotation Guiding Principles[157] zu entnehmen [OAC, 2013].

Das Ergebnis der Arbeit des OAC ist das Open Annotation Data Model[158] (OADM), Version 1. Im Annotieren, welches als ein Vorgang der Erstellung von Assoziationen zwischen verschiedenen Stücken von Information gesehen werden kann, fehlt es oft an einer strukturierten Vorgehensweise. Das OADM bietet ein erweiterbares, interoperables und standardisiertes Ressource Description Framework um Annotationen auszudrücken, so dass diese einfach zwischen verschiedensten Systemen geteilt werden können. Hierbei ist beachten, dass geteilte Annotationen in existierende Sammlungen integriert und wiederverwendet werden können, ohne dass ein Informationsverlust entsteht. Eine Annotation wird als ein Set von verbundenen Ressourcen betrachtet, hier body und target, in der Annahme dass die Bodyressource auf die Targetressource bezogen ist und führt zu einem Basismodell[159] (siehe Abbildung 3-1), vgl. OAC (2013B).

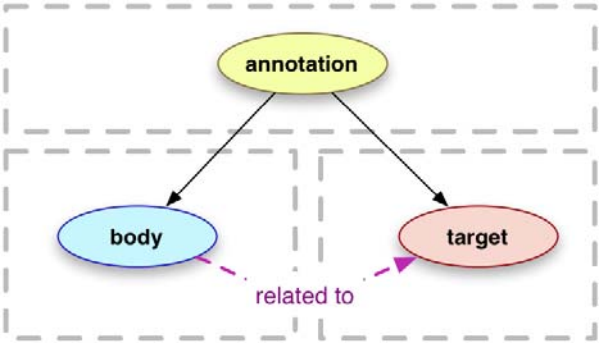


Abbildung 3-1: OA Body und Target [OAC, 2013C]

Im Allgemeinen beschreibt eine Annotation die Beziehung zwischen zwei oder mehreren Ressourcen, sowie dessen Metadaten in einem RDF Graphen. Typischerweise wird eine Annotation durch einen einzigen body, welcher ein Kommentar oder die andere beschriebe Ressource ist, und ein einziges taget beschrieben, über den der body etwas aussagt. Diese Aussagen können weiter präzisiert oder erweitert werden, z.B. Klassifizierung und Identifizierung, oder Motivationen. Der body kann jedem möglichen Mediantyp angehören und jede Art von Inhalt enthalten. Des Weiteren sollte der body und das target über eine HTTP URI identifiziert werden können. Annotationen sind Instanzen der Klasse oa:Annotation. Das Modell stellt keine Klassen für body und target bereit, anstelle diesem werden die 'relationships' ia:hasBody und oa:hasTarget eingeführt, welche durch eine Annotation verknüpft sind, vgl. Abbildung 3-2 und Tabelle 3-1) [OAC, 2013D].

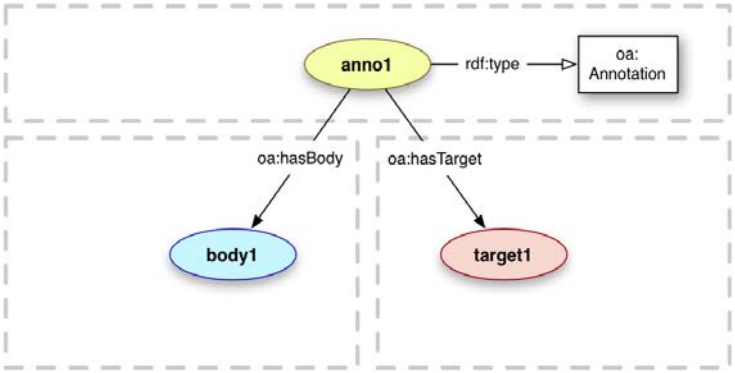
[157] vgl. http://www.openannotation.org/documents/GuidingPrinciplesRevised.pdf (abgerufen 03.12.2013)

[158] vgl. http://www.openannotation.org/spec/core (abgerufen 03.12.2013)

[159] Das vollständige Modell bietet noch weitere Funktionalitäten, welche der Spezifikation zu entnehmen sind.

Tabelle 3-1: Open Annotation Body und Target [OAC, 2013D]

Vokabular	Typ	Beschreibung
oa:Annotation	Class	Klasse der Annotationen. Sie muss mit einer Annotation verknüpft sein.
oa:hasBody	Relationship	Beziehung zwischen der Annotation und dem body der Annotation. Es können 0 bis n Beziehungen bestehen.
oa:hasTarget	Relationship	Beziehung zwischen der Annotation und dem target der Annotation. Es müssen 1 bis n Beziehungen bestehen.

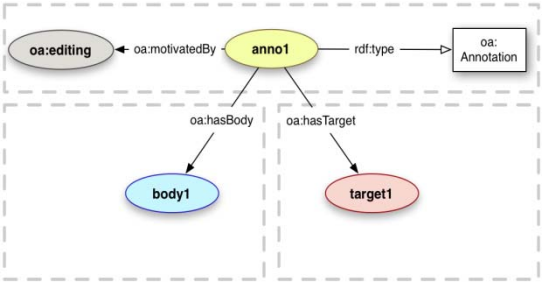



```
<anno1> a oa:Annotation ;
        oa:hasBody <body1> ;
        oa:hasTarget <target1> .
```

Abbildung 3-2: OAC mit Annotation [OAC, 2013E]

Um nachverfolgen zu können, wer wann eine Annotation getätigt hat, kann die 'Annotation Provenance' aufgezeigt werden. Mit Hilfe der Relationships und Propertys oa:annotatedBy, oa:annotatetAt, oa:serializedBy und oa:serializedAt kann sowohl der Urheber, als auch das Datum der Annotation gespeichert werden. Genaueres ist der OADM Kapitel 2 zu entnehmen.

In vielen Fällen ist es zudem wichtig zu verstehen, aus welchen Gründen (Motivationen) eine Annotation getätigt wurde, nicht nur wer daran beteiligt war. Um diese zu modellieren, wird auf eine SKOS Concept hierarchy zurückgegriffen. Motivationen sind daher SKOS Konzepte und können gegenseitige Beziehungen (z.B. von Motivationen) zwischen verschiedenen Communities besser als eine einfache Klassenhierarchie ausdrücken. Jede Annotation sollte mindestens über eine oa:motivatedBy Beziehung verfügen, welche eine Instanz der Klasse oa:Motivation und damit eine Subklasse des skos:Concept ist (vgl. Abbildung 3-3).




```
<anno1> a oa:Annotation ;
    oa:hasBody <body1> ;
  oa:hasTarget <target1> ;
oa:motivatedBy oa:editing .
```

Abbildung 3-3: OA mit Motivationen [OAC, 2013F]

Ein bekanntes und prominentes Beispiel zur Anwendung von Open Annotation ist das Text-Annotation Tool Pundit[160]: Es ermöglicht das Kommentieren von Texten, das Annotieren und das Teilen dieser neu erzeugten Informationen (vgl. Abbildung 3-4 und Abbildung 3-5)

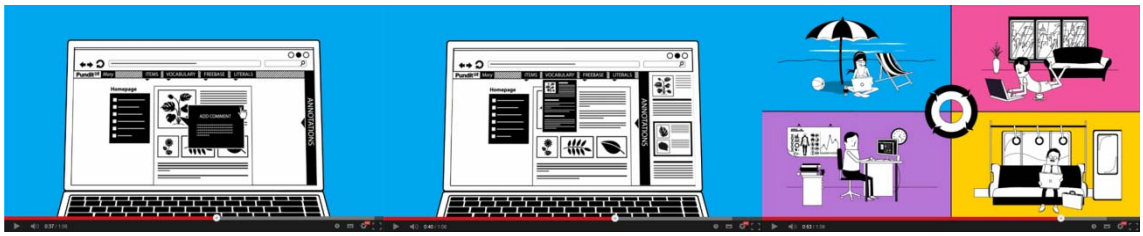


Abbildung 3-4: Pundit Funktionen [PUNDIT, 2013A]

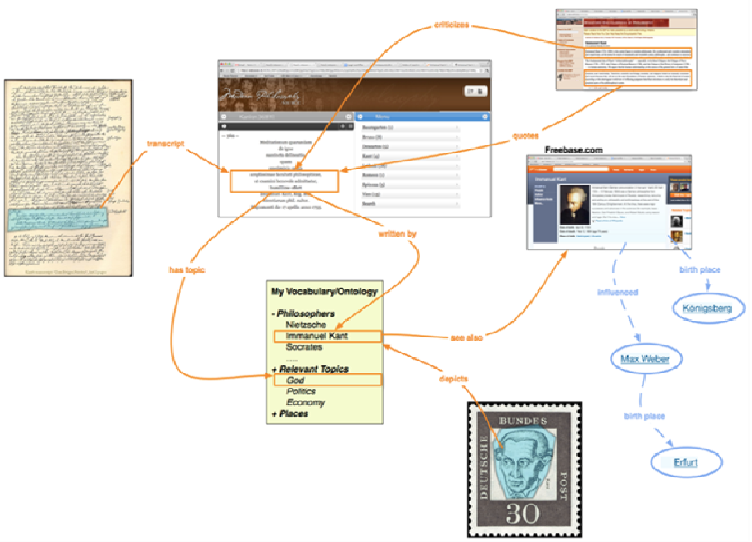


Abbildung 3-5: Pundit Linked Data [PUNDIT, 2013B]

[160] vgl. http://www.thepund.it (abgerufen 03.12.2013)

3.2.2 Vokabularien

Die in Kapitel 3.2.1 beschriebenen Technologien RDFs, SKOS und Open Annotation bilden den Grundstock der in den Geisteswissenschaften genutzten Konzepte der Linked Data. Neben diesen Basistechnologien gibt es jedoch Vokabularien, welche helfen können, Semantik in Fachdaten einzuarbeiten.

Ein wichtiges Hilfsmittel sind Metadaten. Die Dublin Core Metadata Initiative[161] stellt das Vokabular dcterms[162] zur Verfügung. „Dublin Core ist eine Sammlung einfacher und standardisierter Konventionen zur Beschreibung von Dokumenten und anderen Objekten im Internet, um diese mit Hilfe von Metadaten einfacher auffindbar zu machen [WIKIPEDIA, 2013B]". Es bietet u. A. Möglichkeiten Titel (title), Informationen über den Ersteller (creator) und die Beschreibung (description) mit Ressorcen zu verbinden [DCTERMS, 2012].

Linked Data Vokabular Portale wie Linked Open Vocabularies[163] (LOV) geben bereits einen Überblick über bestehende und verbreitete Vokabularien (377, Stand 27.11.2013). Im Fall von LOV sind diese Vokabularien bereits in Kategorien und Sub-Kategorien eingeteilt: City, Library, Where&When, Upper&Meta, Market, Media, Science und Data&Systems. In Bezug auf diese Arbeit sind besonders die Kategorien City (→People), sowie Where&When (→Geography, Geometry) zur Repräsentation von Menschen und geografischen Objekten von Bedeutung (vgl. Abbildung 3-6).

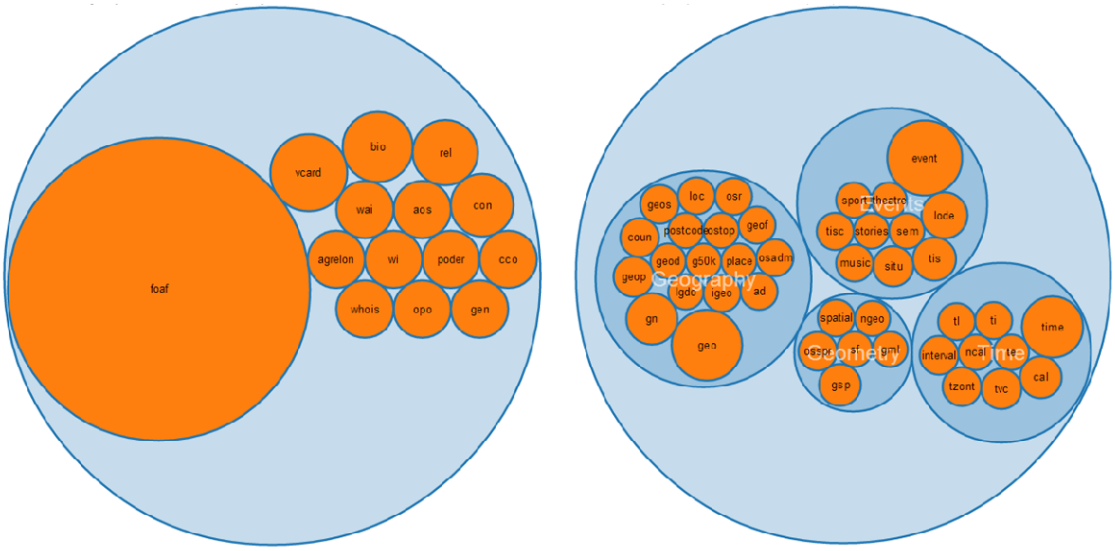


Abbildung 3-6: LOV Vokabulare [LOV, 2013A]

[161] vgl. http://dublincore.org (abgerufen 03.12.2013)

[162] vgl. http://dublincore.org/documents/dcmi-terms (abgerufen 03.12.2013)

[163] vgl. http://lov.okfn.org/dataset/lov (abgerufen 03.12.2013)

Zur Beschreibung von Personen, jedoch nur moderner, nicht historischer Personen, können besonders die Vokabularien FOAF[164], WHOIS[165] und VCARD[166] Eigenschaften aufweisen, die helfen können, auch historischen Personen semantische Linked Data Attribute zuzuweisen.

Das Friend of a Friend - Projekt[167] (FOAF) ist eine einfache Technologie, vor allem zum Teilen von Informationen über Menschen und deren Aktivitäten (wie Fotos und Webblogs). Es erstellt maschinenlesbare Seiten, welche Personen und deren Verbindungen untereinander aufzeigt [FOAF, 2013]. Die FOAF-Spezifikation[168] unterscheidet insbesondere in einen FOAF Core und eine Social Web Extension und bietet eine Vielzahl von Modellierungsmöglichkeiten für Personen, Organisationen, deren Namen und Verbindungen untereinander [BRICKLEY ET AL., 2010]. Vor allem die in VIAF[169] zusammengeschlossenen Bibliotheken nutzen das FOAF Vokabular zur Beschreibung von historischen Personen[170] (Namen). Eine Nutzung von FOAF ist somit auch für historische Personen möglich und ermöglicht den Zugang zu größeren geisteswissenschaftlichen Datenbeständen wie Bibliotheken.

WHOIS[171] beschreibt ein Vokabular, das Personenprofile und Biografien abbilden kann und ist eine Erweiterung zu FOAF. Hierbei können beispielsweise Karrieren (Arbeitsstellen und Tätigkeiten), wie die eines Töpfers dargestellt werden [KANZAKI, 2012].

Die vCard Ontology[172] ermöglicht ebenfalls die Repräsentation moderner Personen als Linked Data, z.B. Gender, TelephoneType, RelatedType [IANELLA ET AL., 2013]. Im Vergleich zu FOAF und WHOIS jedoch, wird sie in keinem größeren gefundenen Datenbestand genutzt, welches gegen eine Modellierung als vCard spricht.

Geometrische Beschreibungen sind unter anderem mit Hilfe der OGC-Vokabularien GeoSPARQL[173], gml[174], sf[175] sowie der GeoVocab-Vokabularien spatial[176] und ngeo[177] möglich.

---

[164] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_foaf.html (abgerufen 03.12.2013)

[165] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_whois.html (abgerufen 03.12.2013)

[166] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_vcard.html (abgerufen 03.12.2013)

[167] vgl. http://www.foaf-project.org (abgerufen 03.12.2013)

[168] vgl. http://xmlns.com/foaf/spec (abgerufen 03.12.2013)

[169] vgl. http://viaf.org (abgerufen 03.12.2013)

[170] Beispiel: http://viaf.org/viaf/100227925/rdf.xml (abgerufen 03.12.2013)

[171] vgl. http://www.kanzaki.com/ns/whois# (abgerufen 03.12.2013)

[172] vgl. http://www.w3.org/TR/vcard-rdf (abgerufen 03.12.2013)

[173] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_gsp.html (abgerufen 03.12.2013)

[174] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_gml.html (abgerufen 03.12.2013)

[175] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_sf.html (abgerufen 03.12.2013)

[176] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_spatial.html (abgerufen 03.12.2013)

[177] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_ngeo.html (abgerufen 03.12.2013)

GeoSPARQL[178] ist eine geographische Abfragesprache des Open Geospatial Consortium[179] (OGC) für Linked Data, die als RDF modelliert sind. GeoSPARQL definiert räumliche Erweiterungen des W3C SPARQL protocol und ermöglicht somit das Abfragen räumlicher Daten des Semantic Web. Es stellt ein grundlegendes Vokabular für Geodaten inkl. deren Lage zur Verfügung. Der GeoSPARQL Standard ist modular aufgebaut: Kernkomponenten (RDFS/OWL Klassen räumlicher Objekte), Geometriekomponenten (RDFs Datentypen für Geometrien), topologische, geometrische Komponenten (definiert topologische Abfragefunktionen), sowie ein topologisches Vokabular (RDF Properties zur Beschreibung von Beziehungen zwischen räumlichen Objekten), vgl. PERRY ET AL., 2012. Beispiele des GeoSPARQL Schemas[180] sind www.geosparql.org zu entnehmen. Das Vokabular gml beschreibt eine Spezialisierung des zuvor beschriebenen GeoSPARQL Standard des OCG zur Definition von Geometriesubtypen[181] wie LineString, Circle oder Point [LOV, 2013B]. Im Gegensatz dazu beschreibt das Vokabular sf[182] Geometrietypen der Simple Feature Spezifikation.

Das OGC stellt jedoch nicht alleine räumliche Vokabulare des Semantic Web zur Verfügung. Als weiterer Anbieter kann GeoVocab[183] gesehen werden. GeoVocab definiert ebenfalls geometrische[184] (geometry) und räumliche[185] (spatial) Vokabularien. Als Basisklasssen können spatial:Feature und spatial:Geometry angesehen werden, welche mit Hilfe der NeoGeo Spatial Ontology Beziehungen untereinander definieren (z.B. overlapps oder connects with), vgl. GEOVOCAB, 2013.

Beschreibungen der Geographie leisten z.B. die Vokabulare geo[186] (WGS84 Geo Positioning) und gn[187] (Geonames Ontology).

Geo[188] dient zur Repräsentation von Positionen im geodätischen WGS84 Datum: Längengrad (longitude), Breitengrad (latitude) und Höhe (altitude). Zwei Klassen (SpatialThing und dessen Subklasse Point) und fünf Properties (latitude, location, longitude, altitude und lat/lon) ermöglichen die Modellierung von Einzelpositionen (Punkten) im globalen Weltkoordinatensystem [W3C, 2009].

GeoNames[189] ist eine freie und kostenlose geografische Datenbank. Sie beinhaltet mehr als 10 Millionen geografische Namen und mehr als 8 Millionen Features, wie z.B. moderne und historische

---

[178] vgl. http://www.opengeospatial.org/standards/geosparql (abgerufen 03.12.2013)

[179] vgl. http://www.opengeospatial.org (abgerufen 03.12.2013)

[180] vgl. http://schemas.opengis.net/geosparql/1.0/geosparql_vocab_all.rdf (abgerufen 03.12.2013)

[181] siehe mehr unter http://schemas.opengis.net/gml/3.2.1/gml_32_geometries.rdf (abgerufen 03.12.2013)

[182] vgl. http://schemas.opengis.net/sf/1.0/simple_features_geometries.rdf (abgerufen 03.12.2013)

[183] vgl. http://geovocab.org (abgerufen 03.12.2013)

[184] vgl. http://geovocab.org/geometry.html (abgerufen 03.12.2013)

[185] vgl. http://geovocab.org/spatial.html (abgerufen 03.12.2013)

[186] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_geo.html (abgerufen 03.12.2013)

[187] vgl. http://lov.okfn.org/dataset/lov/details/vocabulary_gn.html (abgerufen 03.12.2013)

[188] vgl. http://www.w3.org/2003/01/geo/wgs84_pos# (abgerufen 03.12.2013)

[189] vgl. http://www.geonames.org (abgerufen 03.12.2013)

bevölkerte Plätze [GEONAMES, 2013]. Die Repräsentation dieser Datenbank erfolgt mit einer eigenen Ontologie[190]. Geografische Positionen jedoch, werden ebenfalls mit dem bereits gezeigten geo-Vokabular realisiert, vgl. Beispiel Kastell Langenhain[191].

Die bereits vorgestellten Vokabularien zeigen Möglichkeiten der Modellierung im Bereich moderner und historischer Personen, sowie räumlicher Einheiten (Geometrien und Punkte) auf. Die Beschreibung archäologischer Fachdaten können beispielsweise Vokabularien der Heritage Data[192], wie auch des Getty Research Institute[193] (GRI) vereinfachen.

Heritage Data stellt bereits genutzte Thesauri und Vokabularien des Kulturerbes (im Staatsgebiet Großbritanniens) als Linked Open Data zur Verfügung. Die Vokabularien sind in einem RESTful Webservice abrufbar [HERITAGEDATA, 2013A]. Heritage Data nutzt Information von drei verschiedenen Organisationen: English Heritage, Royal Commission on Ancient & Historical Monuments of Scotland (RCAHMS) und der Royal Commission on Ancient & Historic Monuments Wales (RCAHMW), vgl. HERITAGEDATA (2013B). Die Vokabulare sind jeweils als SKOS Konzepte modelliert und decken verschiedenste Bereiche des Kulturerbes ab: z.B. Techniken und Materialen in der Archäologie und Zeitperioden [HERITAGEDATA, 2013C].

Das GRI stellt ebenfalls strukturierte Vokabularien zur Verwendung und zum Abruf bereit: The Art & Architecture Thesaurus (AAT), The Cultural Objects Name Authority (CONA), The Getty Thesaurus of Geographic Names (TGN) und The Union List of Artist Names (ULAN). AAT beinhaltet insbesondere Begriffe, Beschreibungen und andere Informationen generischer Konzepte der Kunst und Architektur. TGN hingegen beinhaltet vor allem Namen und Beschreibungen wichtiger Orte. ULAN deckt den Bereich der Künstler(namen) ab [GETTY, 2013A]. Die Thesauri sind bislang als hierarchische Baumstruktur verfügbar, eine Bereitstellung als Linked Open Data[194] und damit ansprechbare Ressourcen, ist in Arbeit [GETTY, 2013B]. Die in dieser Arbeit verwendeten Fragmente der Samian Ware (South Gallic Terra Sigillata[195]) könnten somit über die AAT-ID 300301427 in naher Zukunft charakterisiert werden.

3.2.3 Linked-Data-Konzepte mit Schwerpunkt Archäologie

Diese Masterarbeit fußt im wesentlich auf drei verschiedenen zu modellierenden Entitäten: Personen (Töpfer), antike Orte (Fund-, Produktionsorte) und Keramikfragmente. Im Bereich der Archäologie gibt es bereits für jede dieser Entitäten Lösungen der Modellierung.

---

[190] vgl. http://www.geonames.org/ontology/documentation.html (abgerufen 03.12.2013)

[191] siehe http://sws.geonames.org/8504432/about.rdf (abgerufen 03.12.2013)

[192] vgl. http://www.heritagedata.org/blog (abgerufen 03.12.2013)

[193] vgl. http://www.getty.edu (abgerufen 03.12.2013)

[194] vgl. http://www.getty.edu/research/tools/vocabularies/Linked_Data_Getty_Vocabularies.pdf (abgerufen 03.12.2013)

[195] vgl. http://getty.edu/vow/AATFullDisplay?find=&logic=&note=&page=&subjectid=300301427 (abgerufen 03.12.2013)

Antike Personen stellen hierbei jedoch eine Ausnahme dar. Wie bereits beschrieben, ist zurzeit kein Linked Data Konzept in den Recherchen ersichtlich, welches darauf ausgelegt ist, historische Personen nur als Person ohne eine Tätigkeit (Event), die mit ihm verbunden ist (z.B. CIDOC CRM), zu modellieren. Eine Möglichkeit ist das bereits beschriebe Konzept der FOAF, an welchem sich z.B. auch der Bibliothekszusammenschluss VIAF bedient.

Pleiades[196] (vgl. Kapitel 7.1) ist ein historischer Gazeteer[197] und ermöglicht es Wissenschaftlern unter Anderem historische geografische Informationen als Linked Data zu erstellen und zu teilen [PLEIADES, 2013A]. Hauptbestandteil ist die Modellierung historischer Orte (Places), dessen Namen und explizit definierten Locations in ihrer jeweiligen Zeit, vgl. Abbildung 3-7 [GILLIES, 2011].

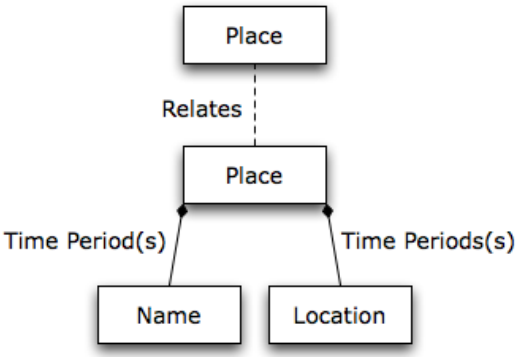


Abbildung 3-7: Pleiades Place Konzept [PLEIADES, 2013B]

Dies schlägt sich ebenfalls in der Ausprägung der Klassen (Place, Location, Name) und Prädikaten (hasLocation, hasName, during, start_date, end_date, has_FeatureType, nameAttested, nameRomanized) wieder [GILLIES, 2013A]. Pleiades stellt weiterhin nützliche Vokabulare[198], zum Beispiel für Zeitperioden[199] und Feature- bzw. Place-Kategorien[200] zur Verfügung. Betrachtet man die RDF-Repräsentation des Places Langenhain[201], sind Verknüpfungen zu anderen bereits vorstellten Konzepten und Vokabularien vorhanden: spatial, foaf, rdfs, rdf, dcterms, geo und skos. Insbesondere die Nutzung bekannter Vokabulare[202] und die Ausrichtung des SKOS Konzepts der Places, Locations und Names auf historische Orte bewirken eine hohe Akzeptanz in den Digital Humanities[203] (siehe auch Projekt Pelagios).

---

[196] vgl. http://pleiades.stoa.org (abgerufen 04.12.2013)

[197] siehe auch http://en.wikipedia.org/wiki/Gazetteer (abgerufen 04.12.2013)

[198] vgl. http://pleiades.stoa.org/vocabularies (abgerufen 04.12.2013)

[199] vgl. http://pleiades.stoa.org/vocabularies/time-periods (abgerufen 04.12.2013)

[200] vgl. http://pleiades.stoa.org/vocabularies/place-types (abgerufen 04.12.2013)

[201] vgl. http://pleiades.stoa.org/places/109100/turtle (abgerufen 04.12.2013)

[202] vgl. https://github.com/isawnyu/pleiades-rdf (abgerufen 04.12.2013)

[203] siehe auch http://www.dig-hum.de (abgerufen 04.12.2013)

Ein Konzept, das alle Komponenten im Bereich des kulturellen Erbes vereint (somit auch Personen, Orte und archäologische Fundstücke), ist CIDOC CRM[204]. Das CIDOC Conceptual Reference Model entspricht zudem dem ISO Standard 21127:2006[205]. „Das CRM ist eine formalisierte Ontologie, mit der unterschiedlich strukturierte Informationen aus dem Bereich des Kulturellen Erbes integriert, vermittelt und ausgetauscht werden können. Das CRM ist der Kulminationspunkt eines über 10 Jahre währenden Prozesses der Standardisierung kultureller Informationen innerhalb des Internationalen Ausschusses für Dokumentation (CIDOC) des Internationalen Museumsrates (ICOM, International Council Of Museums) [CIDOCCRMGMN, 2013]". CIDOC CRM[206] definiert eine Reihe von Klassen[207] und Eigenschaften[208], welchen den gesamten Lebenszyklus z.B. einer Tonscherbe abbilden. Die hohe Komplexität des CIDOC CRM und das damit einhergehende schwierige Mapping eines Sachverhalts (Beispielsweise der Lebenszyklus eines Terra Sigillata Fragments) zeigt die großen Potentiale aber Probleme auf. Des Weiteren ist eine Verbindung zwischen CIDOC CRM und GML möglich [DOERR ET AL, 2013].

3.2.4 Projekte mit Schwerpunkt Archäologie

Das Pelagios Projekt (vgl. Kapitel 7.2) stellt ein großes Repositorium von Verlinkungen verschiedenster mit der Archäologie verbundener Projekte[209] dar. Eine Verknüpfung der verlinkten Datasets mit Pleiades Places erfolgt über Open Annotation (vgl. Kapitel 7.3). Tabelle 3-2 zeigt Pelagios-Projekte, deren Repräsentationsformen und genutzte Konzepte bzw. Vokabulare, welche mit den archäologischen Fachdaten dieser Masterarbeit (Töpfer, Fundorte und Keramikfragmente) zu vergleichen sind.

Tabelle 3-2: Pelagios Datasets und Repräsentationsformen

Projekt	Repräsentation	Konzepte / Vokabulare
CLAROS[210] (Personen, archäol. Objekte, Orte)	HTML / RDF	CIDOC CRM
OpenContext[211] (archäol. Objekte)	HTML / XML / RDF	ArchaeoML CIDOC CRM

[204] vgl. http://www.cidoc-crm.org (abgerufen 04.12.2013)

[205] vgl. http://www.iso.org/iso/catalogue_detail?csnumber=34424 (abgerufen 04.12.2013)

[206] Einführung: http://cidoc-crm.gnm.de/wiki/CIDOC-CRM:Portal (abgerufen 04.12.2013)

[207] vgl. http://cidoc-crm.gnm.de/wiki/Klassen (abgerufen 04.12.2013)

[208] vgl. http://cidoc-crm.gnm.de/wiki/Eigenschaften (abgerufen 04.12.2013)

[209] vgl. http://pelagios.dme.ait.ac.at/api/datasets (abgerufen 04.12.2013)

[210] vgl. http://www.clarosnet.org (abgerufen 04.12.2013)

[211] vgl. http://opencontext.org (abgerufen 04.12.2013)

ARACHNE[212] (archäol. Objekte)	HTML / XML / RDF	dcterms CIDOC CRM TEI XML beliebiges XML
American Numismatic Society[213] / Nomisma[214] (Personen, Münzen, Orte)	HTML / RDF	eigenes Vokabular SKOS dcterms Geo
Finds[215] (archäol. Objekte)	HTML / XML / RDF	beliebiges XML CIDOC CRM
SPQR[216] (Inschriften)	HTML / XML	TEI XML
Ancient World Mapping Centre[217] (Antike Orte)	HTML / JSON / RDF	Open Annotation
Regnum Francorum Online[218] (antike Orte)	HTML	keine
Europeana[219] (archeol. Objekte)	HTML / RDF	SKOS dcterms

Betrachtet man die zuvor vorgestellten Projekte, ist zu erkennen, dass insbesondere im Bereich der archäologischen Objekte das Konzept des CIDOC CRM weit verbreitet ist. Oftmals wird dies mit der Komplexität der Modellierungen und der Möglichkeit alle Sachverhalte in nur einem System und einer Datenbank zu beschreiben, erläutert [ARACHNE, 2013]. Weitere Konzepte und Vokabula-

[212] vgl. http://arachne.uni-koeln.de (abgerufen 04.12.2013)

[213] vgl. http://www.numismatics.org (abgerufen 04.12.2013)

[214] vgl. http://nomisma.org (abgerufen 04.12.2013)

[215] vgl. http://finds.org.uk (abgerufen 04.12.2013)

[216] vgl. http://spqr.cerch.kcl.ac.uk (abgerufen 04.12.2013)

[217] vgl. http://awmc.unc.edu (abgerufen 04.12.2013)

[218] vgl. http://www.francia.ahlfeldt.se (abgerufen 04.12.2013)

[219] vgl. http://www.europeana.eu (abgerufen 04.12.2013)

rien wie SKOS, Open Annotation und dcterms erfreuen sich großen Zuspruchs. Die eigene Modellierung eines archäologischen Sachverhalts sollte daher auf diese bereits vorhandenen Konzepte und Vokabulare bauen. Weitere Projekte, die sich noch nicht an Pelagios beteiligt haben sind Tabelle 3-3 zu entnehmen:

Tabelle 3-3: Archäologische Projekte und deren Repräsentationen

Projekt	**Repräsentation**	**Konzepte / Vokabulare**
STAR / STELLAR (archäol. Sachverhalte)	HTML / RDF	modifiziertes CIDOC CRM SKOS
FACEM (Terra Sigillata Fragmente)	HTML	keine
Lavantine (röm. Keramik)	HTML	keine
DBpedia (Linked Data Repräsentation von Wikipedia-Artikeln)	HTML / RDF	RDF / RDFs FOAF GEO

Artverwandte Projekte wie FACEM[220] und Levantine[221] zeigen leider keine Modellierungen als Linked Data. Das STAR[222]/STELLAR[223] Projekt der Universität Glamorgan setzt auf eine Modifikation des CIDOC CRM. Ein weiteres bekanntes Repositorium ist DBpedia[224]. Dies stellt strukturelle Informationen aus Wikipedia frei zur Verfügung [DBPEDIA, 2013A]. Die daraus resultierenden RDF Repräsentationen[225] zeigen eine hohe Anzahl an bekannten Vokabularien, wie FOAF, GEO oder RDFs. Darüber hinaus baut DBpedia auf schema.org auf, welches ebenfalls Grundstock des bereits vorgestellten RDFa ist [DBPEDIA, 2013B].

[220] vgl. http://facem.at (abgerufen 04.12.2013)

[221] vgl. http://www.levantineceramics.org (abgerufen 04.12.2013)

[222] vgl. http://hypermedia.research.southwales.ac.uk/kos/star (abgerufen 04.12.2013)

[223] vgl. http://hypermedia.research.southwales.ac.uk/kos/stellar (abgerufen 04.12.2013)

[224] deutsche Version: http://de.dbpedia.org (abgerufen 04.12.2013)

[225] z. B. http://de.dbpedia.org/page/Limburgerhof (abgerufen 04.12.2013)

4 Client-Server Architektur

Die Masterarbeit GeInArFa ist eine Client-Server-Anwendung[226]. In Kapitel 8 sind die entwickelten prototypischen Open-Source Clientanwendungen (Map und Explorer) aufgezeigt. Die Verwaltung der Daten, sowie deren Bereitstellung, erfolgt auf einem Server mit Open-Source[227] bzw. kostenlosen Produkten. Die nachfolgenden Kapitel beschreiben die verwendeten Technologien und deren Realisierung (vgl. Kapitel 4.1 und 4.2), sowie die Zusammenarbeit der einzelnen Komponenten (vgl. Kapitel 4.3).

4.1 Server-Technologie

Die Wahl des Betriebssystems des Servers ist entscheidend. Jenes gibt Aufschluss über die zur Verfügung stehenden Softwarekomponenten, die Erweiterbarkeit und dem Grad des Einflusses der Gestaltung der Architektur. CentOS[228] (Community ENTerprise Operating System) ist eine freie Enterprise-class Distribution und zielt darauf ab, 100 prozentig binär kompatibel zu sein [CENTOS, 2013A]. Ziel ist es weiterhin, eine stabile Linux-Lösung für Unternehmen und Einzelpersonen bereitzustellen, ohne starke kommerzielle Unterstützung zu erfahren. Weitere große Ziele sind die einfache Wartung, die Eignung für eine langfristige Nutzung der Produktionsumgebungen, die langfristige Unterstützung des Kerns und die aktive Entwicklung [CENTOS, 2013B]. All diese Ziele entsprechen dem Nutzen, dem GeInArFa verpflichtet ist. Als zentrales Medium der Speicherung der Daten dient die freie PostgreSQL[229] Datenbank (vgl. Kapitel 2.2). Dieses relationale Datenbanksystem kann mit dem PostGIS[230]-Aufsatz auch die Aufgaben der Speicherung von Geometrien und Analyse räumlicher Fragestellungen ausführen und erfüllt hiermit alle Anforderungen der GeInArFa-Infrastruktur. Als Bindeglied zwischen der Datenbank und der Bereitstellung der Geometrien in diversen OGC-Services[231] (vgl. Kapitel 2.1) dient der GeoServer[232] (vgl. Kapitel 2.3). Die Bereitstellung der Geodaten als Web-Map-Service (WMS) oder Web-Feature-Service (WFS) bietet eine große Möglichkeit der Verbreitung im World Wide Web, da sie auf allgemeingültigen Stan-

---

[226] vgl. http://www.e-teaching.org/technik/vernetzung/architektur/client-server (abgerufen 04.12.2013)

[227] vgl. http://de.wikipedia.org/wiki/Open_Source (abgerufen 04.12.2013)

[228] vgl. http://www.centos.org (abgerufen 04.12.2013)

[229] vgl. http://www.postgresql.org (abgerufen 04.12.2013)

[230] vgl. http://postgis.net (abgerufen 04.12.2013)

[231] vgl. http://de.wikipedia.org/wiki/OpenGIS_Web_Service (abgerufen 04.12.2013)

[232] vgl. http://geoserver.org/display/GEOS/Welcome (abgerufen 04.12.2013)

dards[233] aufbauen. Im Falle des WFS ist z.B. eine Repräsentation in GML[234] bzw. GeoJSON[235] möglich, welches Anwendungen, wie beispielsweise Leaflet[236], direkt nutzen können. Basis der Programme, die sowohl die Datenbank, als auch den GeoServer nutzen ist die Programmiersprache JAVA. JAVA ist zudem eine Laufzeitumgebung, die im Jahr 1995 von Sun Microsystems veröffentlicht wurde [JAVA, 2013] und nun dem Hersteller Oracle unterstellt ist [KLEIJN, 2009]. Die Eigenschaften lassen sich kurz zusammenfassen: einfach, objektorientiert, verteilt, sicher, architekturneutral, portabel, perfomant, etc. Im Vergleich zu C++ besteht z.B. keine Notwendigkeit die Memory-Verwaltung im genaueren Blick zu halten. Des Weiteren ist JAVA unabhängig von Prozessor und der Plattform (z.B. Windows vs. Linux) und wird daher als plattformunabhängig bezeichnet [ZAWALSKI, 2013]. Dies kann als wesentlicher Grund aufgeführt werden, weshalb JAVA genutzt wird. Bei einer Serverumstellung (z.B. von Linux auf Windows) könnten die erstellen Programme weiterhin ohne Einschränkungen zum Einsatz kommen. Bei der Auswahl des JDK[237] (Java Development Kit) ist zwischen Oracle JDK[238] und OpenJDK[239] zu unterscheiden. Im Vergleich zum Development Kit von Oracle ist Open JDK Open-Source mit Open-Standards. Oracle JDK basiert jedoch auf OpenJDK Quellcode mit eigenen in sich geschlossenen Komponenten und anderen Lizenzen [OPENJDK, 2010]. Die Wahl des Oracle JDK erfolgt insbesondere aufgrund von Vergleichstests[240] mit dem Geoserver, der auch im Rahmen dieser Arbeit Anwendung findet. Die erstellten Server-Anwendungen sind so genannte Servlets[241]. „Servlets sind Java-Programme, die in einem besonders präparierten Java-Webserver ausgeführt werden. Die Besonderheit daran ist zunächst, dass ein Webserver in Java realisiert werden muss, eine andere Besonderheit die, dass die Java-Programme als Kassen vom Java-Webserver geladen und dort auch verwaltet und mit einer besonderen Servlet-Schnittstelle angesprochen werden. Daher heißt ein Java-Webserver, der Servlets lädt und verwaltet, auch Servlet-Container. Servlets sind somit ein wenig mit Applets vergleichbar. Ein Applet ist ein Java-Programm auf der Clientseite (im Browser), während ein Servlet ein Programm auf der Serverseite (im Server) ist. Der Browser ist der Applet-Container, während der Java-Webserver mit Servlet-Schnittstelle einen Servlet-Container darstellt [ULLENBOOM, 2011]". Beispiele für freie Java-Webserver mit Servlet-Funktionalität sind z.B. Apache[242] Tomcat[243], Jetty[244] oder

---

[233] vgl. http://www.opengeospatial.org/standards (abgerufen 04.12.2013)

[234] vgl. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32554 (abgerufen 04.12.2013)

[235] vgl. http://geojson.org (abgerufen 04.12.2013)

[236] vgl. http://leafletjs.com (abgerufen 04.12.2013)

[237] vgl. http://de.wikipedia.org/wiki/Java_Development_Kit (abgerufen 04.12.2013)

[238] vgl. http://www.oracle.com/technetwork/java/javase/downloads/index.html (abgerufen 04.12.2013)

[239] vgl. http://openjdk.java.net (abgerufen 04.12.2013)

[240] hier insbesondere http://research.geodan.nl/2012/10/openjdk7-vs-oracle-jdk7-with-geoserver und http://superuser.com/questions/593954/performance-oraclejdk-or-openjdk (abgerufen 04.12.2013)

[241] siehe auch http://de.wikipedia.org/wiki/Servlet (abgerufen 04.12.2013)

[242] vgl. http://httpd.apache.org (abgerufen 04.12.2013)

GlassFish[245]. Der Apache Tomcat ist ein Produkt von Apache und steht quelloffen zur Verfügung, ist frei und kann als Standalone-Applikation bzw. in den Apache Server eingebunden werden. Jetty ist ein weiterer HTTP-Server und Servlet-Container unter Apache Lizenz. GlassFish ist die Referenzimplementierung der Java EE 5- und Java EE 6-Spezifikationen [ULLENBOOM, 2011]. Die erstellten Servlets werden in dieser Arbeit mit Bibliotheken und Frameworks versehen. Um die Einbindung dieser zu vereinfachen und Abhängigkeiten zwischen Bibliotheken automatisch zu ermitteln, wird das Tool Maven[246] genutzt. Apache Maven ist ein Software-Projekt-Management-Tool, welches auf dem Project Object Model (POM) basiert [MAVEN, 2013]. Auf die benutzten Bibliotheken und Einbindungen der Abhängigkeiten wird in den entsprechenden Kapiteln eingegangen. Zur Speicherung, der in dieser Arbeit entstandenen Triple, dient ein Sesame[247] Triplestore (vgl. Kapitel 2.5.5), welcher ebenfalls im Apache-Tomcat-Server zur Verfügung steht.

4.2 Realisierung

Die in Kapitel 4.1 vorgestellten Technologien stellen die Basis des GeInArFa-Servers dar. Der verwendete Server erhält die IP-Adresse http://143.93.114.104 mit dem Servernamen geinarfa.fh-mainz.de. Tabelle 4-1 zeigt die verwendeten Programme und Versionen auf:

Tabelle 4-1: Software des GeInArFa-Servers

System	Version
CentOS	6.4 (final)
JDK	Oracle JDK 1.7.0_21
Apache	2.2.15
Tomcat	7.0.41
GeoServer	2.3.2
PostgreSQL	9.2.6
PostGIS	2.0
Sesame Triplestore	2.7.3

[243] vgl. http://tomcat.apache.org (abgerufen 04.12.2013)

[244] vgl. http://www.eclipse.org/jetty (abgerufen 04.12.2013)

[245] vgl. https://glassfish.java.net (abgerufen 04.12.2013)

[246] vgl. http://maven.apache.org (abgerufen 04.12.2013)

[247] vgl. http://www.openrdf.org (abgerufen 04.12.2013)

Die Linux-Distribution CentOS bietet durch den Linux Kernel und freie Programme eine kostenlose Alternative, z.B. zu Windows Server. Laut W3Techs[248] und einer Analyse der Nutzungsstatistiken und Daten über die Marktanteile von Linux im Web, ist CentOS neben Debian und Ubuntu eine der drei meistgenutzten Linux Distributionen [W3TECHS, 2013]. Abbildung 4-1 (links) zeigt CentOS an Position drei (Stand: 22.10.2013). In der rechten Grafik ist zudem zu erkennen, dass Linux und Windows von vielen Websites genutzt wird und dabei ein mittleres Traffic-Niveau erreicht wird. Für die Anforderungen des GeInArFa-Servers (kostenlos, OpenSource, mittleres Traffic-Niveau) ist CentOS somit ausreichend. Anhang B: Installationshinweise Server beinhaltet eine Übersicht von Links, der installierten Programme sowie Installationsanleitungen und Hinweise.

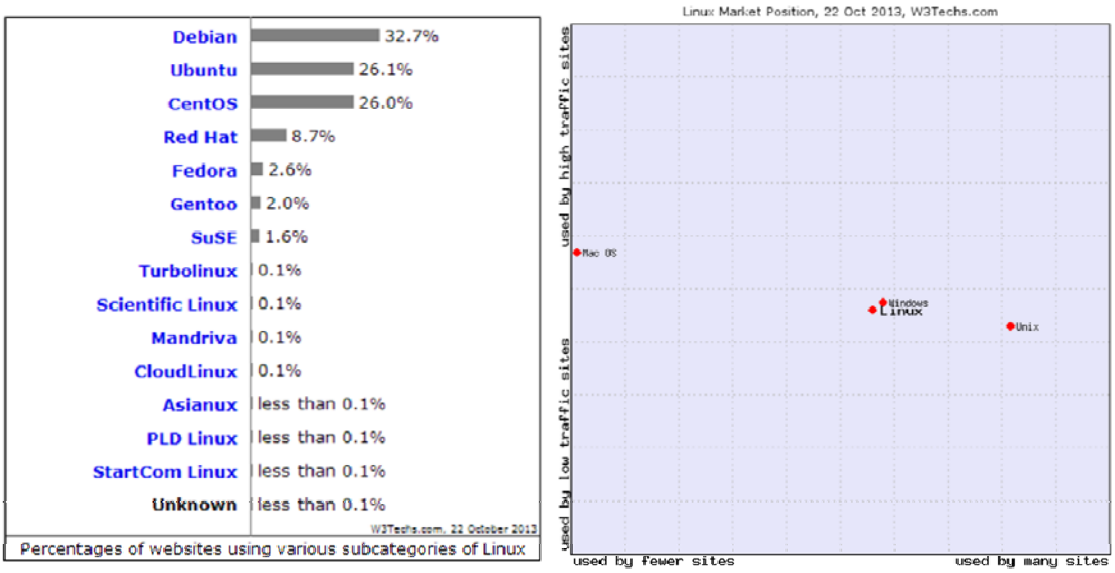


Abbildung 4-1: Linux Distributionen und deren Verbreitung [W3TECHS, 2013]

4.3 Architektur

Die Architektur des GeInArFa-Servers zur Bereitstellung der in Kapitel 5 beschriebenen Ressourcen (Findspots, Fragments und Potters) ist Abbildung 4-2 zu entnehmen. Nach einer Datenaufbereitung, -migration und -versionierung werden jene in einer PostGIS Datenbank gespeichert. Der in den GeInArFa-Server integrierte Geoserver greift auf Tabellen und Views der PostGIS Datenbank zu und stellt diese in OGC-konformen Webdiensten (Kapitel 6.2), hier WMS und WFS, zur Verfügung. Somit ist beispielsweise ein Zugriff auf die Geodaten in den Formaten GML und GeoJSON mittels WFS möglich.

[248] vgl. http://w3techs.com (abgerufen 04.12.2013)

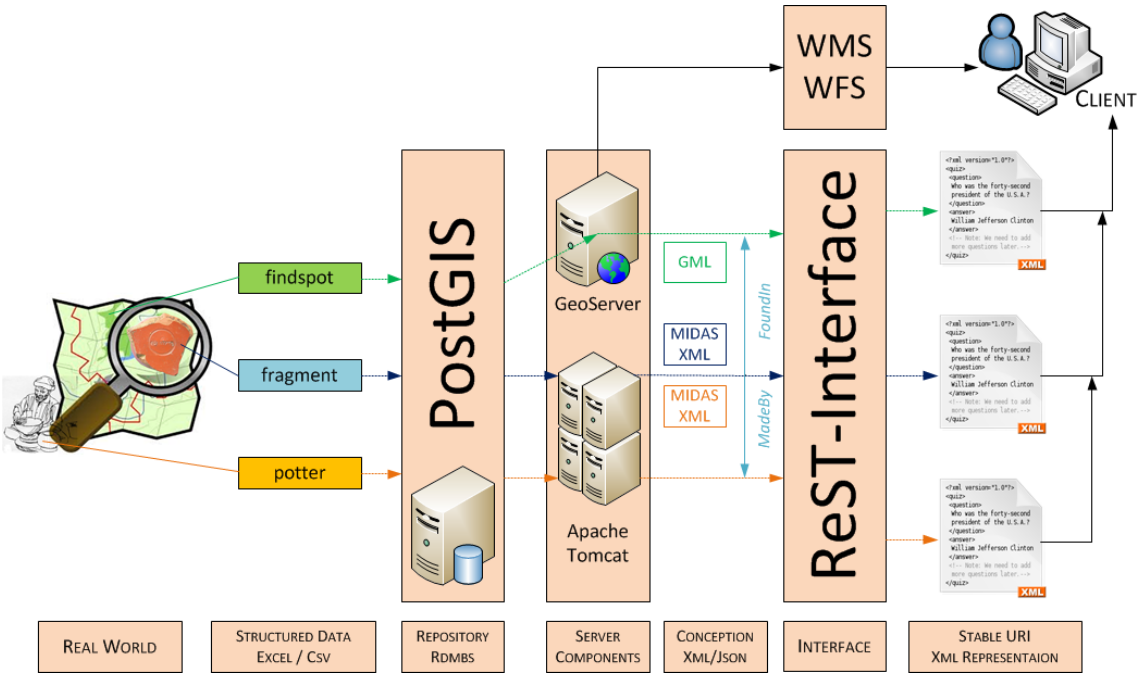


Abbildung 4-2: GeInArFa Serverarchitektur

Apache Tomcat dient als Open Source Webserver und Webcontainer zur Installation von JAVA Servlets[249], welche im Rahmen dieser Masterarbeit als zentrale Server-Anwendungen benutzt werden. Servlets sind eine Java-Plattform-Technologie zur Erweiterung und Verbesserung von Web-Servern. Servlets bieten komponentenbasierte, plattformunabhängige Methoden zur Erstellung von webbasierten Anwendungen. Im Gegensatz zu proprietären Server-Erweiterungsmechanismen (wie Netscape Server API oder Apache-Module) sind Servlets server- und plattformunabhängig (Oracle, 2013[250]). Abbildung 4-3 zeigt ein Diagramm mit einem typischen Ablauf beim Aufruf eines Servlets. Der Geoserver und das ReST-Interface werden als Servlet auf dem Tomcat betrieben.

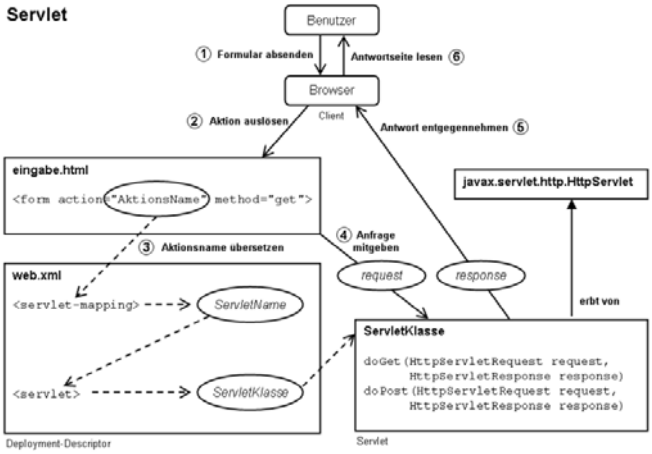


Abbildung 4-3: Aufruf eines Servlets [WIKIMEDIA, 2006]

[249] siehe auch http://de.wikipedia.org/wiki/Servlet (abgerufen 04.12.2013)

[250] Quelle: http://www.oracle.com/technetwork/java/javaee/servlet/index.html (abgerufen 04.12.2013)

Die in Kapitel 6.1 beschriebe ReST-Schnittstelle und deren Bereitstellung von URIs von Ressourcen in diversen Repräsentationen (XML, JSON, etc.), dient als zweites Interface zum Client, neben dem Geoserver. Zudem wird die XML- und JSON-Repräsentation der Findspots über den Geoserver realisiert (vgl. Kapitel 6.1.2).

Die in Abbildung 4-4 aufgezeigte Architektur gibt einen Überblick über die Ermittlung, Bereitstellung und Verknüpfung der Daten mittels RDF-Technologie. Grundlage hierzu bildet ebenfalls die PostGIS Datenbank. In dieser Datenbank erfolgt ebenfalls ein manuelles Mapping zwischen Findspots und Pleiades IDs (vgl. Kapitel 7). Das Servlet zur Bereitstellung des ReST-Interface dient zudem als Erstellungswerkzeug der RDF-Repräsentationen von Findspots, Potters und Fragments. Dabei werden auch RDF-Verknüpfungen zwischen den Ressourcen erzeugt und die Ergebnisse des manuellen Mapping zu Pleiades[251] via ReST zur Verfügung gestellt (vgl. Kapitel 6.1.4) um eine Verlinkung zu PELAGIOS[252] zu gewährleisten. Des Weiteren bietet der GeInArFa-Server eine API, mit der Findspots, mittels RDF-Graphen mit PELAGIOS Datasets in Verbindung gebracht und diese somit ausgelesen werden können (vgl. Kapitel 7.4).

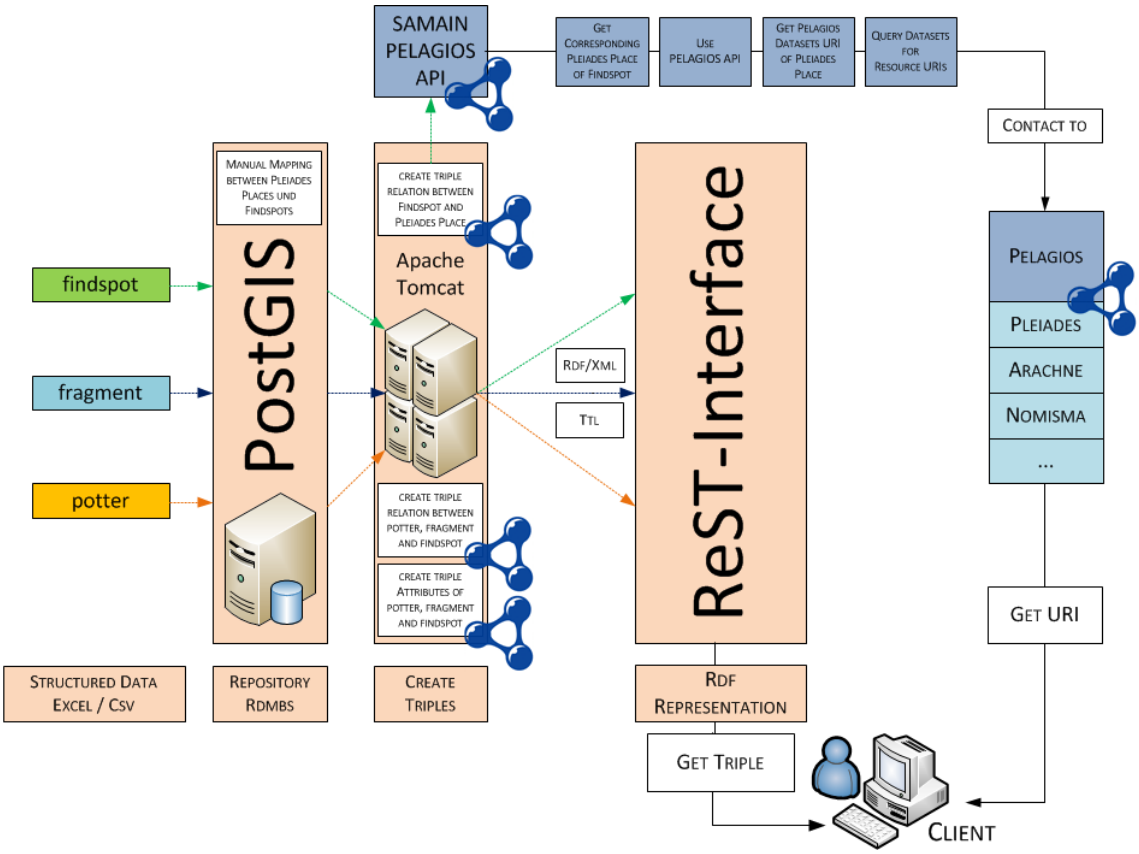


Abbildung 4-4: GeInArFa Serverarchitektur (Linked Data)

[251] vgl. http://pleiades.stoa.org (abgerufen 04.12.2013)

[252] vgl. http://pelagios-project.blogspot.de (abgerufen 04.12.2013)

Sind die Daten mit Hilfe der PostGIS-Datenbank als Linked Data über die ReST-Schnittstelle bereitgestellt und Exportdateien der Verlinkungen erstellt (vgl. Kapitel 6.1.8), werden die so erzeugten Triple im Sesame Triplestore abgelegt. In diesem kann der gesamte Samian-RDF-Graph semantisch abgefragt werden (vgl. Kapitel 6.3) und dient auch als SPARQL-Entry-Point interner und externer Anwendungen, vgl. Abbildung 4-5.

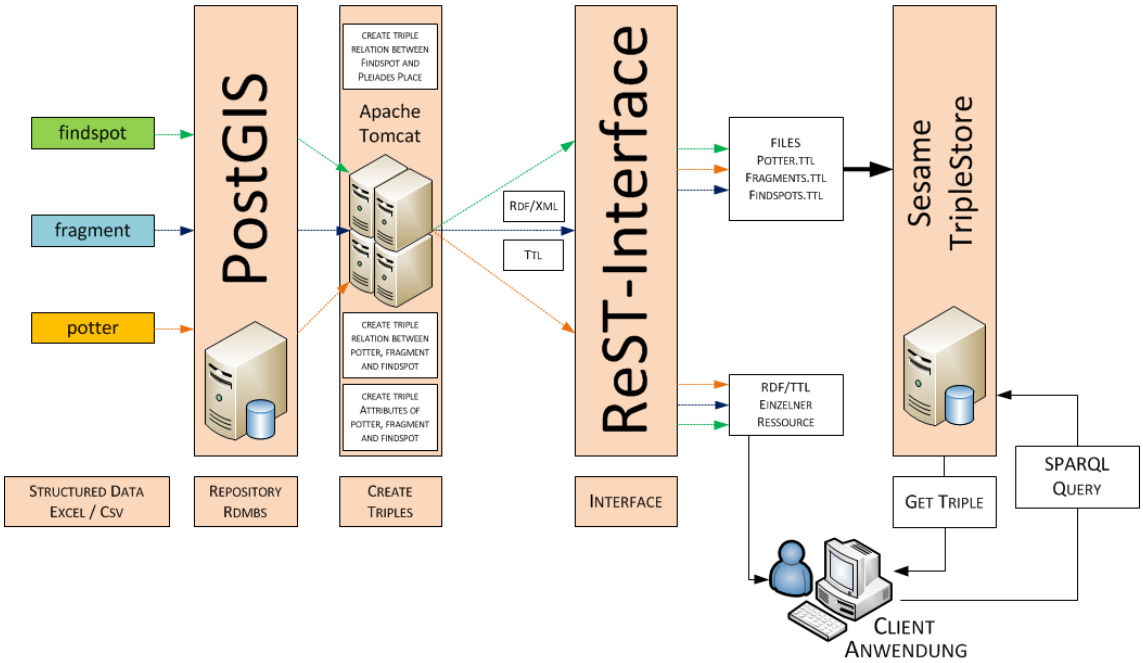


Abbildung 4-5: GeInArFa Serverarchitektur (Triplestore)

4.4 Übersicht der Applikationen

Die in Tabelle 4-2 gezeigten (Servlet-)Applikationen sind unter nachfolgender Basis URL abrufbar. Der Quellcode der einzelnen Apps ist Anhang C: JAVA-Quellcode zu entnehmen.

`http://143.93.114.104/`{application}

Tabelle 4-2: Implementierte Server-Anwendungen

Applikation	Servlet	Funktion
app[253]	kein Servlet nur HTML inkl. JavaScript	Anzeige der prototypischen Anwendungen (Karte, Explorer, ReST-Interface und Time Explorer)
map[254]	kein Servlet	Anzeige der Findspots auf einer

[253] Erläuterung der Funktionsweise in Kapitel 8

	nur HTML inkl. JavaScript	Leaflet-Karte; Test von WMS und WFS
explorer[255]	kein Servlet nur HTML inkl. JavaScript	Anzeige der Linked Data Verbindungen zwischen Potter, Pots und Places, sowie zu Pleiades und Pelagios mit einer Leaflet Karte
rest[256]	rest	Datenbereitstellung der Samian Datenbank in interoperablen Formaten
timeexplorer[257]	getTimeline getRelations getEntities getRDF getTimes getTurtle resetTable sendTriple *Parameter: triple*	prototypische Anzeige einer relativen Zeit-Ontologie als RDF Graph und Google Timeline
getfeature[258]	findspots *Parameter:param*	Proxy zum Erhalt eines GeoJSON Object bei einer WFS- Anfrage an den GeoServer
PelagiosConnectionAPI[259]	PelagiosData *Parameter: PleiadesID, Findspot, type, datasets*	Anzeige der Datasets und URIs zu Pleiades Places, die an Pelagios angeschlossen sind
pleiadesMappingTool[260]	Pleiades	Anzeige der 5 nähesten Punkte der Samian Findspots zu einem Pleiades Places Dump

[254] Erläuterung der Funktionsweise in Kapitel 8.1

[255] Erläuterung der Funktionsweise in Kapitel 8.2

[256] Erläuterung der Funktionsweise in Kapitel 6.1 und 6.1.9

[257] Erläuterung der Funktionsweise in Kapitel 9

[258] Erläuterung der Funktionsweise in Kapitel 6.2

[259] Erläuterung der Funktionsweise in Kapitel 7.4

[260] Erläuterung der Funktionsweise in Kapitel 7.3

4.5 Beurteilung

Die genutzte Serverarchitektur mit den Elementen Apache Tomcat, PostGIS, Geoserver, Sesame Triplestore und diversen eigenen Applikationen (Servlets) zeigt sich als performant und arbeitet einwandfrei. Insbesondere die Konzentration auf eine Programmiersprache (JAVA), in den einzelnen Komponenten (Geoserver, Sesame, Servlets) vereinfacht das Verständnis des komplexen Sachverhalts enorm. Der Geoserver ermöglicht die Veröffentlichung der Daten in Formaten, welche in der Welt des GIS bekannt und verbreitet sind und arrangiert so eine breit gefächerte Nutzung der Daten. Zur Bereitstellung der Daten wird ein ReST-Interface mit diversen Bibliotheken (z.B. Jersey und JAXB genutzt. In zukünftigen Projekten sollte jedoch ein Server genutzt werden, der direkt auf eine Bereitstellung von Daten als Linked Data ausgelegt ist, z.B. Virtuoso.

5 Datenaufbereitung und Datenmigration

Dieses Kapitel beschreibt die Aufbereitung der in dieser Arbeit benutzten Daten (Fragmente von Terra Sigillata Scherben, deren Fundorte und Töpfer der römischen Epoche) und deren Migration in eine PostGIS-Datenbank, welche mit Hilfe einer Versionierung mit Zeitstempeln zur Nachvollziehbarkeit beiträgt. Die Daten sind bereits in einer geschützten Onlineplattform verfügbar[261] (vgl. Abbildung 5-1), entsprechen daher jedoch nicht den in Kapitel 2.5.2 beschrieben Standards der 5 Star Data[262]. Auf den Aspekt der Bereitstellung der Daten als 4 bzw. 5 Star Data geht Kapitel 6 näher ein. Die PostGIS Datenbank dient als zentraler Zugriffspunkt der ReST-Schnittstelle, sowie des Geoservers. Mit der Prozessierung von OGC-konformen Geometrien und der Möglichkeit der räumlichen Analyse jener, bietet diese Datenhaltung über die Aufgaben und Aspekte dieser Arbeit hinaus weitere Ansätze der Forschung. Daher ist eine pragmatische, wie auch effiziente Speicherung der Daten in einem passenden Datenmodell die Grundvoraussetzung des Arbeitens.

Abbildung 5-1: Onlineplattform Samian Ware des RGZM [RGZM, 2013]

[261] vgl. http://www.rgzm.de/samian/home/frames.htm (abgerufen 04.12.2013)

[262] vgl. http://5stardata.info (abgerufen 04.12.2013)

5.1 Ausgangsdaten und Datenaufbereitung

Den Daten der Samian Ware Onlineplattform des RGZM liegt eine Sammlung diverser Forscher und auch einer Mircrosoft Access Datenbank von Danell/Mess zu Grunde. In dieser Datenbank werden insbesondere folgende Entitäten verwaltet:

- Töpfer (Potter) der römischen Epoche
- Fundorte von Terra Sigillata Scherbenfragmenten (inkl. genauen Koordinatenangaben)
- Produktionsstätten von Terra Sigillata
- Terra Sigillata Fragmente (inkl. Verweis auf den Abdruck eines Stempels eines Töpfers [die], sowie die Angabe des Gefäßtyps [potform])

Die vorlegende Datenbank ist in den letzten Jahrzehnten stetig gewachsen und enthält keine normalisierte Datenstruktur, sowie redundante Daten. Auf Grund dieser Tatsache werden die Daten durch das RGZM (durch spezielle SQL-Abfragen in der Access-Datenbank von Mitarbeitern des RGZM) in einer definierten Datenstruktur (welche direkt an die Datenstruktur in der neuen normalisierten PostGIS Datenbank angelegt ist) bereitgestellt. Die Daten liegen in einzelnen Exceltabellen[263] vor und können nach einer weiteren Bearbeitung als CSV[264] Datei gespeichert und in die PostGIS Datenbank eingelesen werden und dienen als Schnittstelle zur Datenbank. In den jeweiligen Tabellen sind bereits Verknüpfungen von Fragmenten zu Töpfern und Fundorten durch den Namen als Schlüssel gegeben. Abbildung 5-2 zeigt ein Beispiel einer Exceldatei von Scherbenfragmenten.

	A	B	C	D	E	F
1	ID	potternamе	die	potform	number	SiteAndFindspot
2	14065	L. A- L-	1a	37	1	London
3	14069	L. A- L-	1a	37	1	Nijmegen|VN
4	14103	Abalanis (Aballanis)	1a	33	1	London
5	14108	Abalanis (Aballanis)	3a	33	1	Heddernheim
6	14109	Abalanis (Aballanis)	3a	33	1	London
7	14121	Abalanis (Aballanis)	4a	33	1	Caerleon
8	14126	Abascantus	1a	32	1	Geislingen
9	14143	Abbo	1a	Dish	1	Rottenburg
10	14170	Abbo	3a	32	1	Saalburg
11	14195	Abbo	3f	31R	1	Cirencester
12	14207	Abbo	4a	31R	1	Silchester
13	14208	Abbo	4a	32	1	Great Chesters
14	14210	Abbo	4a	Dish	1	Heidelberg
15	14240	Abilus	1a	27?	1	London
16	14241	Abilus	1a	33	1	Corbridge

Abbildung 5-2: Exceldatei mit Scherbenfragmenten

[263] Exceltabellen siehe in Anhang M: Sonstiges

[264] CSV Dateien siehe Anhang M: Sonstiges

In Tabelle 5-1 sind die Attribute aufgezeigt, die den einzelnen Entitäten zugeordnet werden können:

Tabelle 5-1: Entitäten und Attribute der Datengrundlage

Entität	**Attribute**
Potter	• Name (eines Töpfers, bzw. einer Organisation)
Findspot	• Name • Koordinaten (Latitude und Longitude, WGS84) • Produktionszentrum (kilnsite) • Datum (Minimum und Maximum)
Fragment	• ID • Töpferstempel (die) • Gefäßform (potform) • Anzahl der gefundenen Fragmente • Fundort • Töpfer, dessen Stempel identifiziert wurde

Die Entität des Findspots ist besonders zu beachten. Im Attribut des Namens sind zwei Phänomene zu beobachten, z.B. Burghöfe und Burghöfe|Geschirrdepot. Hierbei kommt dem Symbol | eine besondere Bedeutung zu. Fundorte können in zwei Bereiche gegliedert werden: Sites und Findspots. Sites können hier als größere räumliche Einheit verstanden werden, welche durch ein Koordinatenpaar (Dezimaltrennzeichen Punkt) repräsentiert werden. Jeder Site können mehrere Findspots zugeordnet werden, die als kleine räumliche Einheit einen expliziten Ort in der Site beschreiben und durch das Symbol | repräsentiert werden, bzw. eine exaktere Angabe liefern. In den folgenden Kapiteln wird eine Umwandlung des Symbols in __ erfolgen um eine Konformität mit Symbolen in URIs zu gewährleisten. Diese Art der Kategorisierung entspricht ebenfalls der Einteilung des Projekts Pleiades, welche Places (als übergeordnete Einheit) in kleinere explizite Einheiten (locations) einordnet. Des Weiteren ist jedem Findspot ein Zeitraum mit absoluten Daten zugeordnet. Diese Daten dienen in erster Linie als Startpunkt für die in Kapitel 9 beschriebene relative Chronologie, um zeitliche Folgen losgelöst von den teils ungenauen absoluten Datumsangaben zu beschreiben. Hierbei ist zu beachten, dass die Zahl 10000 als ein Platzhalter dient, der ein Feld ohne Eingabe kennzeichnet. Ein Findspot kann zudem auch ein Produktionszentrum (kilnsite) für Terra Sigillata gewesen sein bzw. im gegenteiligen Fall eine Exportstädte. In Abbildung 5-3 sind die in dieser Arbeit verwendeten Produktionsstädten aufgezeigt. Ausgangsdaten und diverse Karten von Produktionsstädten sind Anhang D: Kilnsites – Karten und Ausgangsdaten zu entnehmen.

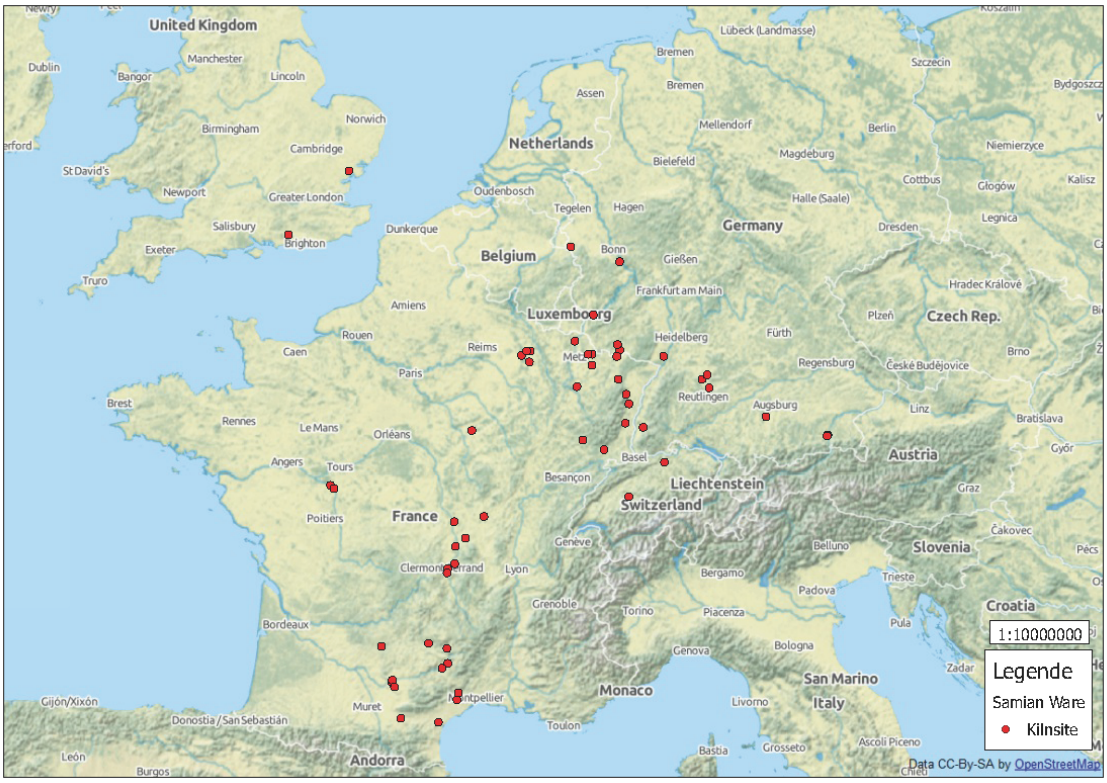


Abbildung 5-3: Kilnsites

Fragmenten sind jeweils der Töpfer und der Fundort zugeordnet. Als besondere Attribute können der Stempelabdruck (die), der jedem einzelnen Töpfer zugeordnet wird, und der Typus (potform) betrachtet werden. Diese liegen in einer nicht normalisierten Form vor. Dies resultiert aus einem fehlenden einheitlichen Katalog von Stempeln der Töpfer und Formen von Terra Sigillata. In nachfolgenden Forschungen sollten solche Kataloge entweder entwickelt, bzw. eingebunden werden.

Ein Problem bei der Datenhaltung können Umlaute und Sonderzeichen sein. Diese können evtl. mit dem Standard UTF-8 behoben werden. Da die Bezeichnungen jedoch als Ressourcenbezeichnungen in der URI benutzt werden ist davon abzusehen. In einem Post-Processing werden diese Umlaute ersetzt (der ursprüngliche Wortlaut wird nicht gespeichert; dies sollte in nachfolgenden Projekten behoben werden) und ggf. falsch gesetzte Leerzeichen, Kommata, Klammern, Fragezeichen und Überschriften entfernt, wie auch das Zeichen | durch __ ausgetauscht. Diese Datei kann nun als kommatrennte Datei (Trennzeichen Semikolon) im Format CSV abgespeichert werden und dient als Grundlage für die in Kapitel 5.3 beschriebenen SQL Skripte.

Betrachtet man die vorherigen Ausführungen, ist zu erkennen, dass ein einfaches Datenmodell ableitbar ist (vgl. Abbildung 5-4). Fragmente besitzen genau einen Fundort und einen Töpfer (links). Ein Fundort kann sowohl ein Place (größere unstrukturierte räumliche Einheit) oder auch eine Location (kleiner expliziter Fundort) sein, wobei ein Place aus mehreren Locations bestehen kann (rechts). Dieser Ansatz kann ebenfalls in der PostGIS Datenbank abgebildet werden und wird in Kapitel 5.2 erläutert.

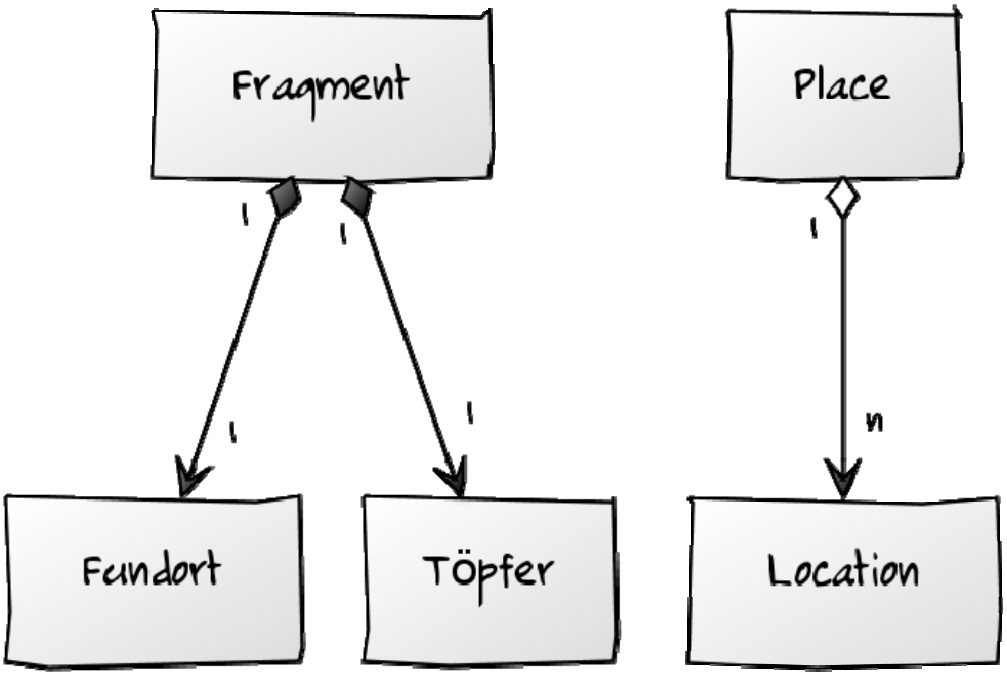


Abbildung 5-4: Datenmodell

5.2 Datenmodell in PostGIS

Die in Kapitel 5.1 erläuterten Strukturen der CSV-Dateien und der daraus ableitbaren Datenbankstruktur findet im PostGIS-Datenmodell Anwendung. Abbildung 2-7 zeigt den schematischen Aufbau, die nachfolgenden SQL-Statements geben den Ist-Zustand im Schema public wider:

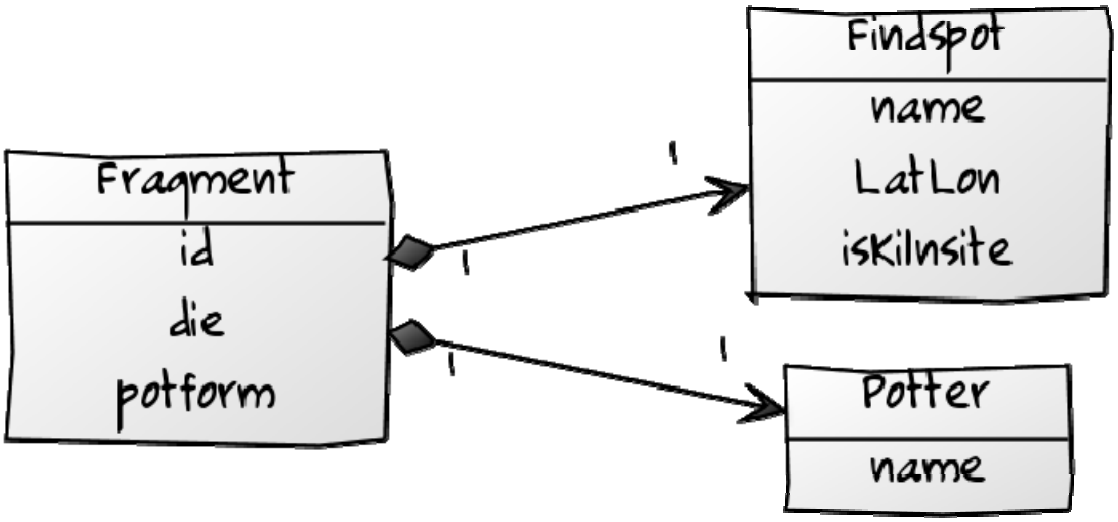


Abbildung 5-5: PostGIS-Datenmodell

```
-- Create Table potter in Schema public
CREATE TABLE public.potter (
   id integer PRIMARY KEY,
   name character varying UNIQUE
);


-- Create Table findspot in Schema public
CREATE TABLE public.findspot (
   id integer PRIMARY KEY,
   name character varying UNIQUE,
   lat double precision,
   lon double precision,
   datemin integer,
   datemax integer,
   kilnsite boolean,
   geom geometry(Point,4326)
);


-- Create Table fragment in Schema public
CREATE TABLE public.fragment (
   id integer PRIMARY KEY,
   p_id integer REFERENCES public.potter(id),
   f_id integer REFERENCES public.findspot(id),
   die character varying,
   potform character varying,
   number integer
);
```

Alle Tabellen sind mit Primärschlüsseln (PRIMARY KEY) ausgestattet (interne ID, bzw. tatsächliche ID der fragments) und beinhalten die in Kapitel 5.1 erläuterten Attribute. Bei einem Update der Daten[265] ist nach diesem Modell sicherzustellen, dass der Name des Töpfers, bzw. des Findspots nicht verändert wird (UNIQUE). Ein Fragment muss auf einen Töpfer, bzw. Findspot referenzieren, der in der Datenbank vorhanden ist (z.B. REFERENCES public.potter(id)). Zudem werden die Koordinaten des findspot in einer OGC-konformen Geometrie abgespeichert[266].

Zur besseren Verarbeitung werden die Daten in drei verschiedenen Schemata gespeichert: import, intern und public (vgl. Abbildung 5-6). Dies ist eine Anlehnung an die ANSI-SPARC-Architektur[267] (auch Drei-Schema-Architektur), welche eine grundlegende Trennung verschiedener konzeptioneller Ebenen anstrebt.

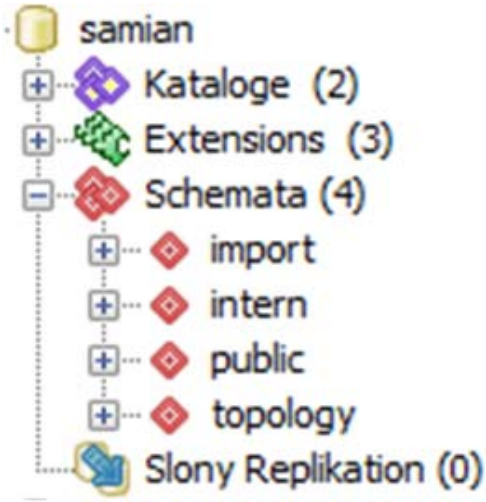


Abbildung 5-6: Datenbankschemata (Überblick)

Das Schema import (vgl. Abbildung 5-7, links) dient zur Eins zu Eins Ablage der CSV-Dateien in die entsprechenden Tabellen. Zudem wird durch Hinzufügen der Spalte importdate eine Versionierung mit Hilfe des SQL-Befehls now()[268] erzeugt (z.B. "2013-06-28 13:22:18.722769+02"). Im Schema intern (vgl. Abbildung 5-7, Mitte) erfolgt ein Kopieren der Einträge des Schemata import, ein evtl. updaten von Informationen (z.B. auch die Erzeugung von OGC-konformen Geometrien) und die Erzeugung von Fremdschlüsseln zwischen Fragmenten, Töpfer und Fundorten, anhand des Namens als Schlüsselfeld. Tabellen des Schema public (vgl. Abbildung 5-7, rechts) werden der Öffentlichkeit zur Verfügung gestellt und entspringen dem Schema intern. Sie sind Bindeglied zwischen den JAVA-Anwendungen (z.B. ReST-Schnittstelle) und dem Geoserver.

---

[265] vgl. Kapitel 5.3

[266] vgl. Kapitel 5.3

[267] vgl. http://dbs.uni-leipzig.de/buecher/DBSI-Buch/HTML/kap1-3.html (abgerufen 04.12.2013)

[268] vgl. http://www.postgresql.org/docs/8.0/static/functions-datetime.html [timestamp with time zone]

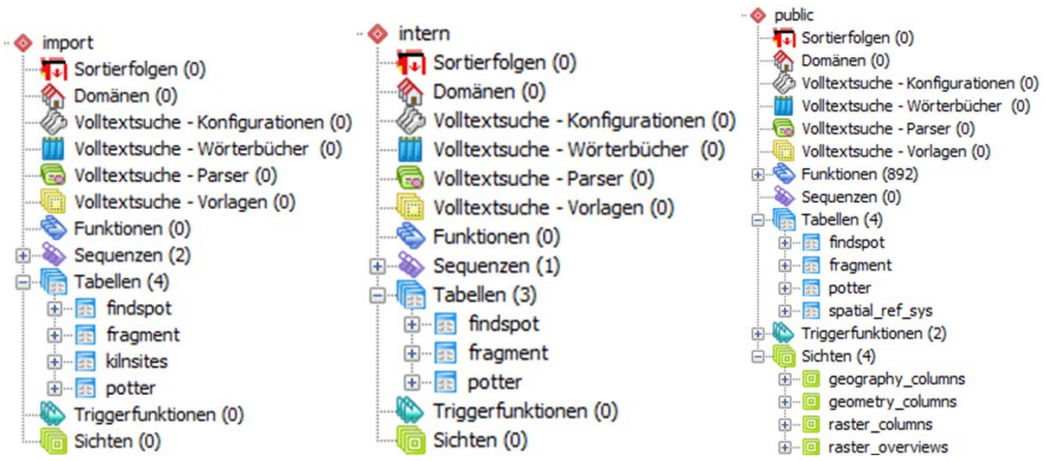


Abbildung 5-7: Datenbankschemata

Um das in Kapitel 7 beschriebene Mapping zwischen findspots und Pleiades IDs zu ermöglichen, wird die Tabelle pleiadesmatch angelegt, welche eine einfache Kreuztabelle darstellt. Diese Tabelle kann durch das Skript 5_create_pleiadesmatch.sql erstellt und das Skript 6_import_pleaidesmatch.sql befüllt werden (vgl. auch Kapitel 7).

```
-- Create Table pleiadesmatch in Schema public
CREATE TABLEpleiadesmatch  (
   findspotcharacter varying PRIMARY KEY,
   pleiadesID character varying UNIQUE
);
```

5.3 Datenüberführung

Die Datenüberführung der in Kapitel 5.1 erläuterten CSV-Dateien in die in Kapitel 5.2 gezeigten PostGIS-Datenstruktur in drei Schemata erfolgt mit Hilfe diverser SQL-Skripte (vgl. Anhang E: SQL-Skripte). Mit Hilfe von PSQL können komplette Skripte auf dem Server ausgeführt werden:

```
samian# SET client_encoding to "UTF-8";
samian# \i '<pfad>.sql'
```

Die folgende Tabelle (vgl. Tabelle 5-2) zeigt die benutzen Skripte und deren Funktionen:

Tabelle 5-2: SQL-Skripte und deren Aufgaben

SQL-Skript	**Funktionen/Aufgaben**
1_init.sql	• Erzeugung des Schema import und intern • Erzeugung derTabellen potter, findspot, fragment inkl. Attribute im Schema intern und public
2_import.sql	• Erzeugung der Tabellen potter, fragment, findspot in import • Import der CSV-Dateien potter, fragment, findspot in die Datenbank • Hinzufügung eines Timestamps • Transformation der Daten in die Tabellen des Schema intern • Erzeugung von Geometrie • Erzeugung von Fremdschlüsseln in der Tabelle fragment • Transformation der Daten in Tabellen in Schema public
3_updateXY.sql	• Erzeugung der inputTabelle in import • Import der CSV mit Timestamp • Kopieren der neuen Daten (wenn Name/ID nicht vorhanden) in Tabelle in Schema intern bzw. Update wenn Daten verändert wurden • Löschung der Daten in den Tabellen in Schema public • Tabellen in Public und Referenzierung neu erstellen
4_delete.sql	• Löschung der Schemata import und intern • Löschung aller Tabellen

Beispielhaft seien an dieser Stelle die SQL-Statements zum Import der CSV-Dateien (hier potter.txt, je nach Speicherort und Name muss dieser Teil des Skriptes verändert werden) die Erstellung eines Timestamps und die Prozessierung von OCG-konformen Geometrien (hier POINT, WGS84) aufgezeigt.

```
-- Import potter Table with DELIMITER ';' AND Create Timestamp
SET client_encoding = 'win1252';
COPY import.potter FROM '/tmp/potter.txt' WITH DELIMITER AS ';';
ALTER TABLE import.potter ADD COLUMN importdate timestamp with time zone;
UPDATE import.potter SET importdate = now();

-- Add and set Geometry Column
SELECT AddGeometryColumn( 'intern', 'findspot', 'geom', 4326, 'POINT', 2 );
UPDATE intern.findspot SET geom = ST_SetSRID( ST_Point(lon, lat),4326 );
```

Das Skript 3_addkilnsite.sql kommt zum Einsatz, um der Tabelle findspot, Produktionszentren, welche in einer anderen Datei gespeichert sind, hinzuzufügen. Mit diesen Update-Skripten kann zudem eine Änderung der Attribute (nicht der Name, da dieser als Schlüssel zwischen den Entitäten fungiert) erfolgen.

Nach Ausführung der Skripte 1, 2 und 3 ist die Datenüberführung der CSV-Dateien in die PostGIS Datenbank abgeschlossen. Die Tabellen des Schema public können nun von diversen Anwendungen (z.B. JAVA und Geoserver) benutzt werden.

Zur einzelnen Darstellung von Places oder Locations werden im Skript init zwei Views (vgl. nachfolgende SQL Statements) im Schema public angelegt (locations und places).

```
CREATE VIEW locations AS SELECT * FROM public.findspot
WHERE public.findspot.name LIKE '%\_\_%' ORDER BY public.findspot.name ASC;

CREATE VIEW places AS SELECT * FROM public.findspot

WHERE public.findspot.name NOT LIKE '%\_\_%' ORDER BY public.findspot.name ASC;
```

Diese Views werden vom GeoServer angesprochen, damit eine Darstellung von Places und Locations in der Map (vgl. Kapitel 8.1) gewährleistet ist.

5.4 Beurteilung

Die vorliegende PostGIS Datenbank kann für alle wesentlichen, in dieser Arbeit genutzten, Anwendungen genutzt werden. Sie ist sowohl einfacher Datenspeicher (Tabellen), Geometrieablage, und Verbindungsglied zum Geoserver. Durch die Geometrieerweiterung sind zudem räumliche Abfragen möglich, welche in dieser Arbeit jedoch nicht verwendet werden. Die verwendete Datenstruktur erfüllt ihren Zweck. Eine Erweiterung dieses Modells um eine eigene Entität Potform und Die ist jedoch anzustreben. So kann eine weitergehende Klassifizierung der Keramikfragmente erfolgen. Das verwendete 3-Schemata Modell hat sich ebenfalls als nützlich erwiesen. Das einfache Hinzufügen von Daten mittels kommagetrennter Textdatei und SQL Skripten lässt Flexibilität zu. Als kritisch ist jedoch anzumerken, dass die Datengrundlage des RGZM (CSV-Datei) zunächst noch manuell bearbeitet werden muss. Dieser Schritt sollte in kommenden Versionen ebenfalls automatisch erfolgen. Darüber hinaus kann eine Trennung zwischen Site und Findspot (Place und Location) auch in den Datenbanktabellen (nicht nur durch die Zeichen __) als sinnvoll erachtet werden und die Flexibilität des Systems erhöhen.

6 Datenbereitstellung

Zur Bereitstellung von Datenauszügen aus der in Kapitel 5 beschriebenen PostGIS Datenbank dienen zwei verschiedene Systeme: Das ReST-Schnittstelle, sowie die OCG- konformen Webdienste WMS[269] und WFS[270] (vgl. Kapitel 2.1). Über die ReST-Schnittstelle ist ein hierarchisches Durchsuchen der Ressourcen und die Ansicht jener in verschiedenen Formaten möglich. Die OGC-Webdienste WMS und WFS repräsentieren die Fundort-Geometrien und deren Attribute. In den nachfolgenden Kapiteln werden der Aufbau und die Möglichkeiten der ReST-Schnittstelle und die gewählten Repräsentationsformen der Ressourcen aufgezeigt. Des Weiteren wird die Umsetzung und Implementierung der ReST-Schnittstelle, sowie die Einrichtung des Geoservers[271] zur Ausgabe der OGC-Webdienste, erläutert. Um die Modellierungen zu konkretisieren wird auf ein Beispiel zurückgegriffen, das in Abbildung 6-1 aufgezeigt ist und auf folgende URIs verweist:

- http://143.93.114.104/rest/samian/potters/Balbinus
- http://143.93.114.104/rest/samian/fragments/30170
- http://143.93.114.104/rest/samian/findspots/Langenhain

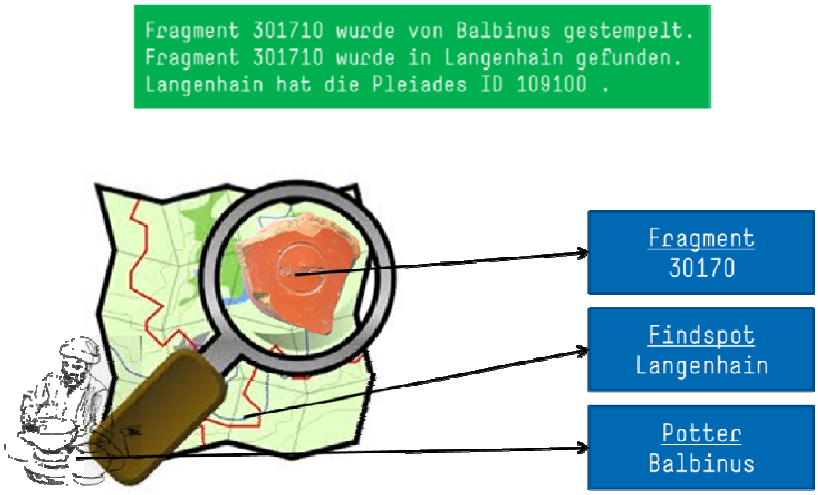


Abbildung 6-1: Datenbereitstellung (Beispiel)

[269] vgl. http://www.opengeospatial.org/standards/wms (abgerufen 04.12.2013)

[270] vgl. http://www.opengeospatial.org/standards/wfs (abgerufen 04.12.2013)

[271] vgl. http://geoserver.org (abgerufen 04.12.2013)

6.1 Die ReST-Schnittstelle

Die ReST-Schnittstelle ist hierarchisch aufgebaut, kann gemäß der Spezifikationen (vgl. Kapitel 2.4) in den Formaten XML[272], JSON[273] und HTML[274] abgerufen werden und dient zur Bereitstellung eindeutiger URIs[275]. In dieser Arbeit wird der Zugriff auf HTTP-GET[276] Methoden beschränkt, somit werden nur Daten vom Server abgerufen, selbst aber keine neuen Daten gesendet. Als zentraler Entry-Point dient die URL: http://143.93.114.104/rest. Im Browser liegt dem Benutzer nach Aufruf des Entry-Points eine HTML-Repräsentation der Schnittstelle vor (vgl. Abbildung 6-2). Die Repräsentation wird durch Links unterstützt, so dass auch im Browser ein hierarchisches Durchsuchen möglich ist. Die Schnittstelle ist dem DIO ReST-Interface[277] nachempfunden, welche in einer hierarchischen Struktur Inschriftenkataloge in XML (aktuell TEI EpiDoc[278]) zur Verfügung stellt.

Abbildung 6-2: ReST Entry Point (HTML)

Eine ReST-Schnittstelle besitzt jedoch den Anspruch sowohl menschen- als auch maschinenlesbar zu sein. Ein automatisches Parsen[279] der HTML-Seiten ist sowohl zeitintensiv, wie auch mit erheblichen Problemen verbunden, sollte sich die HTML-Repräsentation ändern. Die semantische Auszeichnung[280], z.B. mit einem <address>-Tag[281], kann das Parsen vereinfachen, wird jedoch in die-

[272] vgl. http://www.w3.org/XML (abgerufen 04.12.2013)

[273] vgl. http://json.org/json-de.html (abgerufen 04.12.2013)

[274] vgl. http://www.w3.org/html (abgerufen 04.12.2013)

[275] vgl. http://de.wikipedia.org/wiki/Uniform_Resource_Identifier (abgerufen 04.12.2013)

[276] vgl. http://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol#HTTP_GET (abgerufen 04.12.2013)

[277] vgl. http://www.inschriften.net/rest (abgerufen 04.12.2013)

[278] Stand 26.10.2013

[279] vgl. http://de.wikipedia.org/wiki/Parser (abgerufen 04.12.2013)

[280] vgl. z.B. http://www.vorsprungdurchwebstandards.de/theory/semantischer-code (abgerufen 04.12.2013)

ser Arbeit nicht verwendet. Ein maschinenlesbarer Zugriff kann über die Steuerung des Accept-Headers[282] erfolgen. Der Entry-Point und die hierarchischen Elemente können als XML und JSON empfangen werden. Zur Steuerung des Accept-Headers kann z.B. eine Scriptsprache wie JavaScript[283] in Verbindung mit jQuery[284] oder auch ein in den Mozilla Firefox integriertes Add-On (RESTClient[285]) benutzt werden.

Abbildung 6-3 zeigt ein Beispiel für eine XML-Repräsentation des Entrypoints. Die ReST-Schnittstelle wird durch einen <collections>-Tag repräsentiert. In dieser Masterarbeit steht nur eine Datenbank und Collection (genannt samian) zur Verfügung. Jedes Unterelement der Collections wird durch eine eindeutige ID (name oder id) und Link (href) identifiziert. Das Attribut href ermöglicht das automatisierte Durchsuchen der ReST-Schnittstelle mit einer Maschine, da der Zielort hiermit bekannt ist. Die Collection samian enthält drei verschiedene Ressourcen-Typen (vgl. Kapitel 5 und Abbildung 6-4): Potters, Findspots und Fragments.

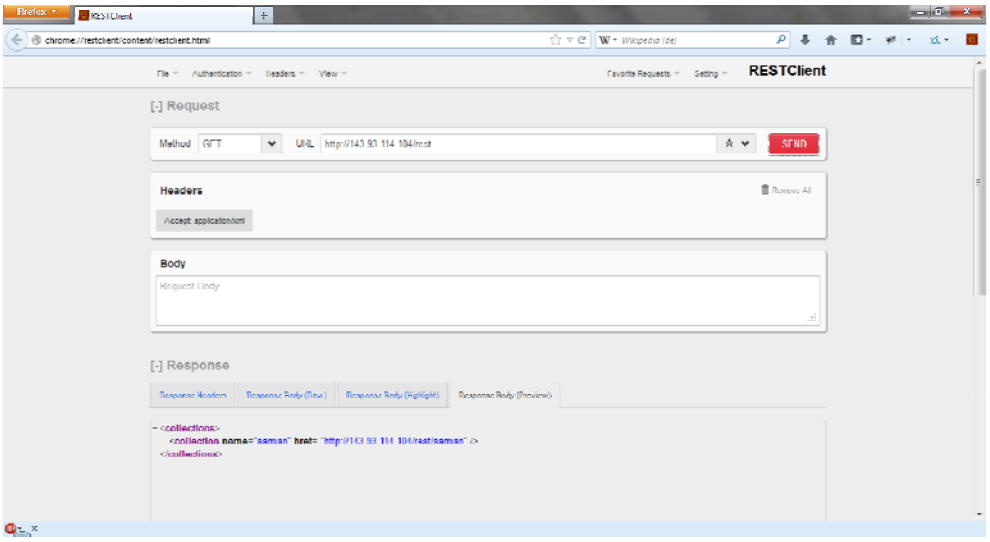


Abbildung 6-3: Schnittstelle mit ReST-Client (Firefox)

Eine Auflistung der Ressourcen ist über die nachfolgenden URLs möglich. Hierbei ist zu beachten, dass eine große Anzahl von Ressourcen zu Problemen, z.B. in der Anzeige im Browser oder beim Auslesen mit jQuery, führen kann. Hier sind evtl. besondere Auslesestrategien zu verwenden. Aufgrund der hohen Anzahl von Fragmenten ist die Repräsentation als HTML-Seite nicht möglich.

---

281 vgl. http://www.w3schools.com/tags/tag_address.asp (abgerufen 04.12.2013)

282 vgl. http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html (abgerufen 04.12.2013)

283 vgl. http://de.wikipedia.org/wiki/JavaScript (abgerufen 04.12.2013)

284 vgl. http://jquery.com (abgerufen 04.12.2013)

285 vgl. https://addons.mozilla.org/de/firefox/addon/restclient (abgerufen 04.12.2013)

- Potters: http://143.93.114.104/rest/samian/potters
- Findspots: http://143.93.114.104/rest/samian/findspots
- Fragments: http://143.93.114.104/rest/samian/fragments

```
1.  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2.  <collection id="samian" href="http://143.93.114.104/rest/samian">
3.    <resources id="potters" href="http://143.93.114.104/rest/samian/potters"/>
4.    <resources id="findspots" href="http://143.93.114.104/rest/samian/findspots"/>
5.    <resources id="fragments" href="http://143.93.114.104/rest/samian/fragments"/>
6.  </collection>
```

Abbildung 6-4: ReST Repräsentation (XML)

Abbildung 6-5 zeigt einen Auszug einer XML-Auflistung der in der Datenbank gespeicherten Findspot-Ressourcen. Auf die Repräsentationsformen der drei Ressourcen-Typen wird in den nachfolgenden Kapiteln eingegangen.

```
1.  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2.  <collection id="samian" href="http://143.93.114.104/rest/samian">
3.    <resources id="findspots" href="http://143.93.114.104/rest/samian/findspots">
4.      <resource id="Aardenburg" href="http://143.93.114.104/rest/samian/findspots/Aardenburg"/>
5.      <resource id="Aislingen" href="http://143.93.114.104/rest/samian/findspots/Aislingen"/>
6.      <resource id="Altenstadt" href="http://143.93.114.104/rest/samian/findspots/Altenstadt"/>
7.      <resource id="Aulnay__fortress" href="http://143.93.114.104/rest/samian/findspots/Aulnay__fortress"/>
8.      <resource id="Baden-Baden" href="http://143.93.114.104/rest/samian/findspots/Baden-Baden"/>
9.      <resource id="Bainbridge" href="http://143.93.114.104/rest/samian/findspots/Bainbridge"/>
10.     <resource id="Baldock" href="http://143.93.114.104/rest/samian/findspots/Baldock"/>
11.     <resource id="Bendorf" href="http://143.93.114.104/rest/samian/findspots/Bendorf"/>
12.     <resource id="Benwell" href="http://143.93.114.104/rest/samian/findspots/Benwell"/>
13.     <resource id="Bickenbach" href="http://143.93.114.104/rest/samian/findspots/Bickenbach"/>
14.     <resource id="Birrens" href="http://143.93.114.104/rest/samian/findspots/Birrens"/>
15.     <resource id="Brigetio" href="http://143.93.114.104/rest/samian/findspots/Brigetio"/>
16.     <resource id="Broomholm" href="http://143.93.114.104/rest/samian/findspots/Broomholm"/>
17.     <resource id="Budapest" href="http://143.93.114.104/rest/samian/findspots/Budapest"/>
18.     <resource id="Budapest__Aquincum,_Hoard" href="http://143.93.114.104/rest/samian/findspots/Budapest__Aquincum,_Hoard"/>
19.     <resource id="Burghoefe" href="http://143.93.114.104/rest/samian/findspots/Burghoefe"/>
20.     <resource id="Burghoefe__Geschirrdepot" href="http://143.93.114.104/rest/samian/findspots/Burghoefe__Geschirrdepot"/>
```

Abbildung 6-5: XML-Repräsentation der Findspots

6.1.1 Soll-Eigenschaften der Ressourcen

Betrachtet man die drei vorgestellten Entitäten (vgl. Kapitel 5) Potter, Findspot und Fragment stellt sich die Frage welche Eigenschaften einen Töpfer, einen Fundort und ein Scherbenfragment zugeschrieben werden können um sie eindeutig zu identifizieren.

Tabelle 6-1 zeigt die in dieser Arbeit zugrundeliegenden Eigenschaften. Dementsprechend sind die in den nachfolgenden Kapiteln beschriebenen Repräsentationsformen zu wählen, damit eine bestmögliche Abdeckung der Soll-Eigenschaften erreicht wird.

Tabelle 6-1: Solleigenschaften der Ressourcen

Entität	Eigenschaften
Potter	• ist historische Person • hat Namen • hat Beruf → Töpfer • ist ggf. eine Organisation • lebte in einer Zeitepoche • Geschlecht

Findspot	• existiert in zwei Formen:
	  o findspot (großräumliches Objekt, abstrakt)
	  o site (konkreter Fundort)
	• hat deutschen Namen
	• hat Koordinaten (lat/lon) → Punktgeometrie
	• ist ggf. Produktionsstätte für Töpferarbeiten(ja/nein)

Fragment	• ist archäologisches Fundstück (Fragment eines Gegenstandes) • entstand in der Zeit der Römer • ist von Menschen hergestellt • ist Terra Sigillata (erstellt aus Ton) • hat Attribute: Stempelabdruck, Potform

6.1.2 HTML Repräsentationen der Ressourcen

Als Standard-Repräsentation der Schnittstelle ist das HTML-Format vorgesehen. Somit ist sichergestellt, dass ein Benutzer bei einem Zugriff mit einem handelsüblichen Browser eine von Menschen lesbare Antwort des Servers erhält. Eine Beispielansicht für das Fragment 30170[286] ist Abbildung 6-6 zu entnehmen. Hierbei sind sowohl die hierarchische Struktur (s.o.: Collection, Resources, Ressource) als auch Attribute des jeweiligen Elements (mittlerer Bereich) gezeigt. Der Footer dient zur Auswahl weiterer Repräsentationsarten des Elements, wie XML, JSON, Turtle und RDF/XML (vgl. nächste Kapitel). Diese können mit Hilfe anderer Werkzeuge (z.B. JavaScript) maschinell ausgelesen werden.

[286] vgl. http://143.93.114.104/rest/samian/fragments/30170 (abgerufen 04.12.2013)

Abbildung 6-6: Fragment 30170 (HTML)

6.1.3 XML Repräsentationen der Ressourcen

Im Vergleich zur HTML-Repräsentation kann die XML-Ansicht des jeweiligen Elements nicht direkt mit dem Browser angezeigt werden. Für einen vereinfachten Zugriff wird die Schnittstelle {resource}.xml[287] zur Verfügung gestellt. Mit einer zusätzlichen Information (Accept: application/xml) im Header einer Anfrage (z.B. mit JavaScript) kann die XML-Repräsentation jedoch auch aus der direkten URI[288] abgeleitet werden. Dieses Verfahren ist für alle drei Resourcen (Potter, Fragments und Findspots) realisiert und kann analog für JSON Repräsentationen ({resource}.json, Accept: application/json) verwendet werden.

Kapitel 3.1 zeigt mögliche und in anderen Projekten verwendete XML-Repräsentationen wie MIDAS XML, MADS XML, GML, ArchaeoML und TEI auf. Betrachtet man die Töpfermatrize eines Töpfers als Inschrift[289] und unabhängig vom Scherbenfragment, könnte man dies beispielsweise im XML Dialekt TEI[290] ausdrücken. Im Rahmen dieser Masterarbeit werden die Fragmente jedoch als Einheit von Gegenstand mit diversen Merkmalen (die, potform) gesehen. Eine Dokumentation von

[287] z.B. http://143.93.114.104/rest/samian/potters/Balbinus.xml (abgerufen 04.12.2013)

[288] z.B. http://143.93.114.104/rest/samian/potters/Balbinus (abgerufen 04.12.2013)

[289] z.B. Inschrift von DIO: http://www.inschriften.net/rest/di019/articles/di019-0001 (abgerufen 04.12.2013)

[290] vgl. http://www.tei-c.org (abgerufen 04.12.2013)

ArchaeoML (wie z.B. von Open Context genutzt[291]) ist leider nicht mehr verfügbar, sodass jene Repräsentationsform für den hiesigen Sachverhalt nicht anzustreben ist.

MADS (Metadata Authority Description Standard) ist ein Standard[292] der Library of Congress[293] in Washington DC, welche sich selbst als größte Bibliothek der Welt bezeichnet [LOC, 2013]. Durch eine XML Repräsentation der Metadaten nach Bibliotheksstandards könnten sich Synergien zwischen den eigenen geisteswissenschaftlichen Daten und derer aus Nationalbibliotheken ergeben. Ursprünglich ist MADS als Schema für ein authority element set von Bibliotheken gedacht, das Metadaten über agents, events und terms speichert [MADS, 2013]. Bezogen auf diese Arbeit wären hier insbesondere agents wie Personen und Organisationen zur Darstellung von Töpfern aber auch terms wie geografische Elemente zur Darstellung von Findspots von Bedeutung[294]. Abbildung 6-7 zeigt beispielhaft die Darstellung des Töpfers Balbinus[295] in MADS XML. Es ist zu erkennen, dass sowohl der Name (Balbinus), der Beruf (Potter) und das Geschlecht (male) angegeben werden kann. Um die Grundgesamtheit der Entitäten Potter, Findspot und Fragment zu untermauern, ist jedoch ein Standard zu bevorzugen, der sowohl Personen/Organisationen, Objekte und Orte sowie ggf. den Prozess der Entstehung und des Auffindens abbilden kann.

```
<mads:mads version="2.0" xsi:schemaLocation="http://www.loc.gov/mads/v2 http://www.loc.gov/standards/mads/mads-2-0.xsd">
<mads:identifier>Balbinus</mads:identifier>
<mads:authority>
    <mads:name type="personal">
        <mads:namePart type="given">Balbinus-</mads:namePart>
    </mads:name>
</mads:authority>
<mads:extension>
    <mads:profession>
        <mads:professionTerm>Potter</mads:professionTerm>
    </mads:profession>
    <mads:gender>
        <mads:genderTerm>male</mads:genderTerm>
    </mads:gender>
</mads:extension>
</mads:mads>
```

Abbildung 6-7: MADS-Beispiel (Balbinus)

MIDAS XML ist ein W3C[296]-XML Schema, das ein gemeinsames Format für die Speicherung, Verarbeitung und den Austausch von historischen Informationen bereitstellt. Es deckt damit alle gesammelten Informationen des MIDAS-Heritage[297] Standard ab [HERITAGESTANDARDS, 2013]. MIDAS-Heritage ist ein britischer Standard des Kulturerbes, u. A. zur Aufzeichnung von Informationen über Gebäude, archäologischen Fundstätten, und Artefakten. Er wird z.B. von nationalen Re-

[291] vgl. http://opencontext.org/subjects/67BDF2BC-E33C-4D99-9947-5D15D78A45E6 (abgerufen 04.12.2013)

[292] vgl. http://www.loc.gov/standards (abgerufen 04.12.2013)

[293] vgl. http://www.loc.gov/index.html (abgerufen 04.12.2013)

[294] Beispiele siehe http://www.loc.gov/standards/mads/userguide/examples.html (abgerufen 04.12.2013)

[295] vgl. http://143.93.114.104/rest/samian/potters/Balbinus (abgerufen 04.12.2013)

[296] vgl. http://www.w3.org (abgerufen 04.12.2013)

[297] vgl. http://www.english-heritage.org.uk/publications/midas-heritage/midas-heritage-2012-v1_1.pdf (abgerufen 04.12.2013)

gierungsorganisationen in Großbritannien, Organisationen des Kulturerbes in der Forschung und von Unternehmen eingesetzt um einen Austausch von Wissen und eine Langzeitarchivierung zu ermöglichen [ENGLISHHERITAGE, 2013]. MIDAS XML ist ein Ergebnis des FISH (Forum on Information Standards in Heritage) und liegt in der aktuellsten Version 2.0 [HERITAGESTANDARDS, 2013] in diversen Schemata[298] und einer Dokumentation[299] vor. Somit ist eine Modellierung von Personen (actor), Objekten (object) und auch Orten (spatial) möglich[300]. Hierbei gibt die MIDAS XML Schema Documentation[301] (Version 1.1, Änderungen zu 2.0[302] beachten) einen Überblick, der für diese Arbeit relevanten Modellierungen des actor[303], object[304] und spatial[305]. Des Weiteren ist die Einbindung kontrollierter[306] und eigener Vokabulare möglich.

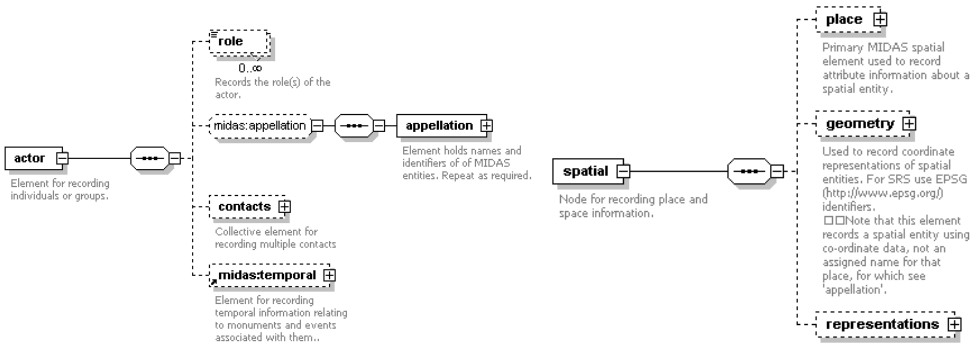


Abbildung 6-8: Modellierung in MIDAS XML [HS, 2013A+B]

Ein actor kann eine bestimmte Rolle (Potter) einnehmen, einen bestimmten Namen tragen (appellation) und seine Tätigkeit in einer bestimmten Zeit (temporal) ausgeübt haben (vgl. Abbildung 6-8, links). Ein räumliches Objekt (spatial) kann durch diverse Eigenschaften (place) wie den Namen und die Adresse sowie eine Geometrie (geometry) dargestellt werden (vgl. Abbildung 6-8, rechts). Ein Objekt[307] kann durch seine Eigenschaften (z.B. Gefäßform, Material, Herstellungsart, Zustand), seinen Töpfer und den Fund- bzw. Herstellungsort beschrieben werden und beinhaltet somit einen actor und ein spatial Element. Hiermit ist die Grundgesamtheit der Entitäten Potter, Findspot und Fragment gewährleistet und eine Abbildung jener in einem einzigen anerkannten

[298] vgl. http://www.heritage-standards.org.uk/midas/schema/2.0 (abgerufen 04.12.2013)

[299] vgl. http://www.heritage-standards.org.uk/midas/docs (abgerufen 04.12.2013)

[300] Beispiele siehe http://www.heritage-standards.org.uk/midas/examples/2-0 (abgerufen 04.12.2013)

[301] vgl. http://www.heritage-standards.org.uk/midas/docs/1-1 (abgerufen 04.12.2013)

[302] vgl. http://www.heritage-standards.org.uk/files/Midas1_1_to_Midas2_0_Change_Log.pdf (04.12.2013)

[303] vgl. http://www.heritage-standards.org.uk/midas/docs/1-1/actor (abgerufen 04.12.2013)

[304] vgl. http://www.heritage-standards.org.uk/midas/docs/1-1/object (abgerufen 04.12.2013)

[305] vgl. http://www.heritage-standards.org.uk/midas/docs/1-1/spatial (abgerufen 04.12.2013)

[306] vgl. http://www.jiscmail.ac.uk/files/FISH/INSCRIPTION_IdentifiersV2.rtf (abgerufen 04.12.2013)

[307] vgl. http://www.heritage-standards.org.uk/midas/docs/1-1/object/midas_object_p2.png (abge. 04.12.2013)

Schemata möglich. In Abbildung 6-9 und Abbildung 6-10 sind mögliche Umsetzungen des MIDAS XML Standards für Potter und Fragments (inkl. der in der Datenbank vorliegenden Eigenschaften und Beziehungen) exemplarisch an den Töpfern Bilbanus[308] und dem Fragment 30170[309] aufgezeigt, welche in der ReST-Schnittstelle implementiert ist, vgl. Kapitel 6.1.9.


```
– <actors schemaLocation="http://www.heritage-standards.org.uk/midas/schema/2.0 http://www.he
  – <actor>
      <role>Potter</role>
    – <appellation>
        <name preferred="id" type="id">Balbinus</name>
        <identifier type="uri">http://143.93.114.104/rest/samian/potters/Balbinus</identifier>
      </appellation>
      <contact/>
    – <temporal>
      – <span>
        – <display>
            <appellation type="period">Roman</appellation>
          </display>
        – <start>
            <appellation type="date" qualifier="circa">-30</appellation>
          </start>
        – <end>
            <appellation type="date" qualifier="circa">300</appellation>
            >
          </end>
        </span>
      </temporal>
    </actor>
  </actors>
```


Abbildung 6-9: Balbinus (MIDAS-XML)


```
– <objects xsi="http://www.w3.org/2001/XMLSchema-instance" schemaLocation="http://www.heritage-standards.org.uk/
  – <object>
    – <appellation>
        <name preferred="id" type="id">30170</name>
        <identifier type="uri" namespace="samian">http://143.93.114.104/rest/samian/fragments/30170</identifier>
      </appellation>
    – <character>
        <objecttype certainty="certain">Terra Sigillata</objecttyp
      – <descriptions>
        – <description>
            <full/>
            <summary/>
          </description>
        </descriptions>
      – <classifications>
          <classification namespace="samian">Potform</classification>
          <classification>doubtful</classification>
        </classifications>
      – <manufacture>
        – <materials>
            <material>Pottery</material>
            <material>Terra Sigillata</material>
          </materials>
          <technique>pottery</technique>
        – <temporal>
          – <span>
            – <display>
                <appellation type="period">Roman</appellation>
              </display>
            – <start>
                <appellation type="date" qualifier="circa">-30</appellation>
              </start>
            – <end>
                <appellation type="date" qualifier="circa">300</appellation>
                >
              </end>
            </span>
          </temporal>
        – <decorations>
            <decoration type="die">1a</decorat
            <decoration>4ie</decoration>
          </decorations>
        </character>
      – <condition>
          <state>good</state>
          <completeness>fragment</completenes
        </condition>
        – <spatial>
          – <place>
            – <address>
                <streetaddres
                <city/>
                <adminarea/>
                <postcode/>
                <country/>
              </address>
            – <namedplace>
                <location type="locality">Langenhain</location>
              </namedplace>
              <gridref/>
              <geopolitical/>
              <cadastral/>
              <directions/>
            </place>
          – <geometry>
              <boundingBox srs="EPSG:4326" minx="50.365103" maxy="8.644794" miny="8.644794" maxx="50.365103"/>
            – <spatialappellation type="centrepoint">
              – <quickpoint>
                  <srs>EPSG:4326</srs>
                  <x>8.933333397</x>
                  <y>51.28333282</y>
                </quickpoint>
              – <entity spatialtype="Point" uri="http://143.93.114.104/rest/samian/findspots/Langenhain" namespace="samian">
                  <wkt srs="EPSG:4326">POINT(50.365103 8.644794)</wkt>
                </entity>
                <capturemethod/>
                <source/>
              </spatialappellation>
            </geometry>
          </spatial>
        – <actor>
            <role>Potter</role>
          – <appellation>
              <name preferred="id" type="id">Balbinus</name>
              <identifier type="uri">http://143.93.114.104/rest/samian/potters/Balbinus</identifier>
            </appellation>
            <contact/>
          – <temporal>
            – <span>
              – <display>
                  <appellation type="period">Roman</appellation>
                </display>
              – <start>
                  <appellation type="date" qualifier="circa">-30</appellation>
                </start>
              – <end>
                  <appellation type="date" qualifier="circa">300</appellation>
                </end>
              </span>
            </temporal>
          </actor>
```


Abbildung 6-10: Fragment 30170 (MIDAS-XML)

[308] vgl. http://143.93.114.104/rest/samian/potters/Balbinus.xml

[309] vgl. http://143.93.114.104/rest/samian/fragments/30170.xml

MIDAS XML ermöglicht ebenfalls die Darstellung räumlicher Objekte. Hierbei wird die Repräsentation der Geometrie als Well-Known-Text (WKT[310]) unter Einbeziehung des EPSG[311] Codes ermöglicht[312] [HERITAGESTANDARDS, 2013B]. In Anbetracht der Tatsache, dass mit den räumlichen Daten ggf. auch in der nachstehenden Analyse mit GIS-Software gearbeitet werden wird, ist ein von (Web-)Geoanwendungen und OGC-konformes unterstütztes Datenformat (vgl. Kapitel 2.1) wie z.B. GML zu bevorzugen. Aus diesem Grund werden Findspots als GML[313] zur Verfügung gestellt. Kapitel 6.2 erläutert die Zusammenarbeit zwischen der benutzten PostGIS Datenbank, dem GeoServer und der damit bereitstellten Webdienste WMS und WFS. Der implementierte Web Feature Service (WFS) bietet bereits die Möglichkeit GML-Objekte des gesamten Findspot-Datenbestandes zu senden. Mit Hilfe von Filtern kann diese Anfrage so modifiziert werden, dass nur einzelne Findspots ausgegeben werden können. Das Ergebnis wird in der XML-Ansicht eines Findspots der ReST-Schnittstelle angezeigt, z.B. Findspot Langenhain[314] (vgl. Abbildung 6-11). Des Weiteren bietet GML die Möglichkeit Semantik einzuarbeiten und damit weitere Informationen zu ggf. anderen Ressourcen innerhalb des XML zur Verfügung zu stellen, vgl. Kapitel 3.


```
- <wfs:FeatureCollection numberOfFeatures="0" timeStamp="2013-11-12T14:59:45.500Z" xsi:schema
  typeName=Samian%3Afindspot http://www.opengis.net/wfs http://143.93.114.104:80/geoserver/schemas/
  - <gml:featureMembers>
    - <Samian:findspot gml:id="findspot.84">
        <gml:name>Langenhain</gml:name>
        <Samian:lat>50.365103</Samian:lat>
        <Samian:lon>8.644794</Samian:lon>
        <Samian:datemin>100</Samian:datemin>
        <Samian:datemax>270</Samian:datemax>
        <Samian:kilnsite>false</Samian:kilnsite>
      - <Samian:geom>
        - <gml:Point srsDimension="2" srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
            <gml:pos>8.644794 50.365103</gml:pos>
          </gml:Point>
        </Samian:geom>
      </Samian:findspot>
    </gml:featureMembers>
  </wfs:FeatureCollection>
```


Abbildung 6-11: Langenhain (GML)

[310] vgl. http://www.opengeospatial.org/standards/sfa (abgerufen 04.12.2013)

[311] vgl. http://www.epsg-registry.org (abgerufen 04.12.2013)

[312] vgl. http://www.heritage-standards.org.uk/midas/examples/2-0/spatial_example.xml (abger. 04.12.2013)

[313] vgl. http://www.opengeospatial.org/standards/gml (abgerufen 04.12.2013)

[314] vgl. http://143.93.114.104/rest/samian/findspots/Langenhain.xml (abgerufen 04.12.2013)

6.1.4 JSON Repräsentationen der Ressourcen

Die in Kapitel 6.1.3 vorgestellten und implementierten XML Schemata bieten keine direkte Transformation des XML Datensatzes in eine äquivalente JSON-Repräsentation an. Die Tate Gallery[315] zeigt jedoch eine Möglichkeit zur Darstellung[316] von Künstlerattributen und deren Werke[317]. Um ein einheitliches Bild der ReST-Schnittstelle zu gewährleisten, wird dieses Schema jedoch nicht angewandt. Potter und Fragmente werden somit in einem beliebigen JSON-Objekt beschrieben, welche nur die originären Werte, der in der PostGIS-Datenbank gespeicherten Daten, widerspiegelt. Abbildung 6-12 zeigt exemplarisch die JSON-Darstellung des Fragments 30170[318] und dessen Potter Balbinus[319].


```
{
    "fragment": {
        "id": "30170",
        "uri": "http://143.93.114.104/rest/samian/fragments/30170",
        "attributes": {
            "material": "pottery",
            "classification": "samian ware",
            "id": "30170",
            "die": "1a",
            "potform": "doubtful",
            "time": {
                "period": "Roman",
                "start": "-30",
                "end": "300"
            }
            "stampedBy": "Balbinus",
            "findspot": {
                "name": "Langenhain",
                "lat": "50.365103",
                "lon": "8.644794"
            }
        }
    }
}
```

```
{
    "potter": {
        "name": "Balbinus",
        "uri": "http://143.93.114.104/rest/samian/potters/Balbinus",
        "attributes": {
            "profession": "Potter",
            "name": "Balbinus",
            "gender": "male",
            "time": {
                "period": "Roman",
                "start": "-30",
                "end": "300"
            }
        }
    }
}
```


Abbildung 6-12: Fragment 30170, Balbinus (JSON)

Zur Darstellung geometrischer Objekte kann GeoJSON[320] genutzt werden. Der implementierte und genutzte Geoserver stellt ebenfalls GeoJSON Objekte via WFS zur Verfügung. GeoJSON ist ein allgemein anerkanntes Format[321] zur Kodierung von geografischen Datenstrukturen. Die in den prototypischen Anwendungen genutzte JavaScript Bibliothek Leaflet nutzt ebenfalls GeoJSON Objekte als WFS-Schnittstelle (vgl. Kapitel 8). GeoJSON Objekte können sowohl eine Geometrie, ein Feature (Geometrie inkl. weiterer Eigenschaften) oder auch eine Gruppe von Features mit den Typen Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon und GeometryCollection beschreiben [BUTLER ET AL., 2008]. Im Fall des Findspot wird das Feature 'Point' verwendet, welches die Geometrie (Koordinaten: Längen-, und Breitengrad) und weitere

---

[315] vgl. http://www.tate.org.uk (abgerufen 04.12.2013)

[316] vgl. http://www.tate.org.uk/about/our-work/digital/collection-data (abgerufen 04.12.2013)

[317] vgl. https://github.com/tategallery/collection (abgerufen 04.12.2013)

[318] vgl. http://143.93.114.104/rest/samian/fragments/30170.json (abgerufen 04.12.2013)

[319] vgl. http://143.93.114.104/rest/samian/potters/Balbinus.json (abgerufen 04.12.2013)

[320] vgl. http://geojson.org (abgerufen 04.12.2013)

[321] vgl. http://wiki.geojson.org/Users (abgerufen 04.12.2013)

Eigenschaften (Name, Produktionsstätte) bereitstellt. Zudem erlaubt GeoJSON optional die Verwendung eines CRS (Coordinate Reference System, vgl. EPSG-Code), als Default wird das WGS84-Datum genutzt [BUTLER ET AL., 2008]. Analog zur Darstellung der GML-Objekte wird zur Erstellung von GeoJSON-Objekten der GeoServer mit entsprechender Filterung genutzt, vgl. Kapitel 6.1.9. Abbildung 6-13 zeigt beispielhaft die Darstellung des Findspots Langenhain[322] als GeoJSON-Feature mit den Koordinaten 8.644794, 50.365103 und den Attributen wie Name und Kilnsite. Zudem ist das CRS (EPSG: 4326) und damit das WGS84-Datum definiert.


```
{
    "type": "FeatureCollection",
    "features":
    [
        {
            "type": "Feature",
            "id": "findspot.84",
            "geometry":
            {
                "type": "Point",
                "coordinates":
                [
                    8.644794,
                    50.365103
                ]
            },
            "geometry_name": "geom",
            "properties":
            {
                "name": "Langenhain",
                "lat": 50.365103,
                "lon": 8.644794,
                "datemin": 100,
                "datemax": 270,
                "kilnsite": false
            }
        }
    ],
    "crs":
    {
        "type": "EPSG",
        "properties":
        {
            "code": "4326"
        }
    }
}
```


Abbildung 6-13: Langenhain (GeoJSON)

6.1.5 RDF Repräsentationen der Ressourcen

Das Potential von Linked Data in der Archäologie, insbesondere im Bereich der Samian Ware, kann nur durch eine bestmögliche semantische Modellierung der einzelnen Entitäten und deren Beziehungen gezeigt werden. Kapitel 3 gibt einen Überblick über Vokabularien und Lösungen anderer Projekte, welche im Kontext dieser Masterarbeit von Relevanz sind. Die Generierung von Interope-

[322] vgl. http://143.93.114.104/rest/samian/findspots/Langenhain.json (abgerufen 04.12.2013)

rabilität in RDF setzt einige Überlegungen[323] voraus, welche in Abbildung 6-14 verdeutlicht werden.

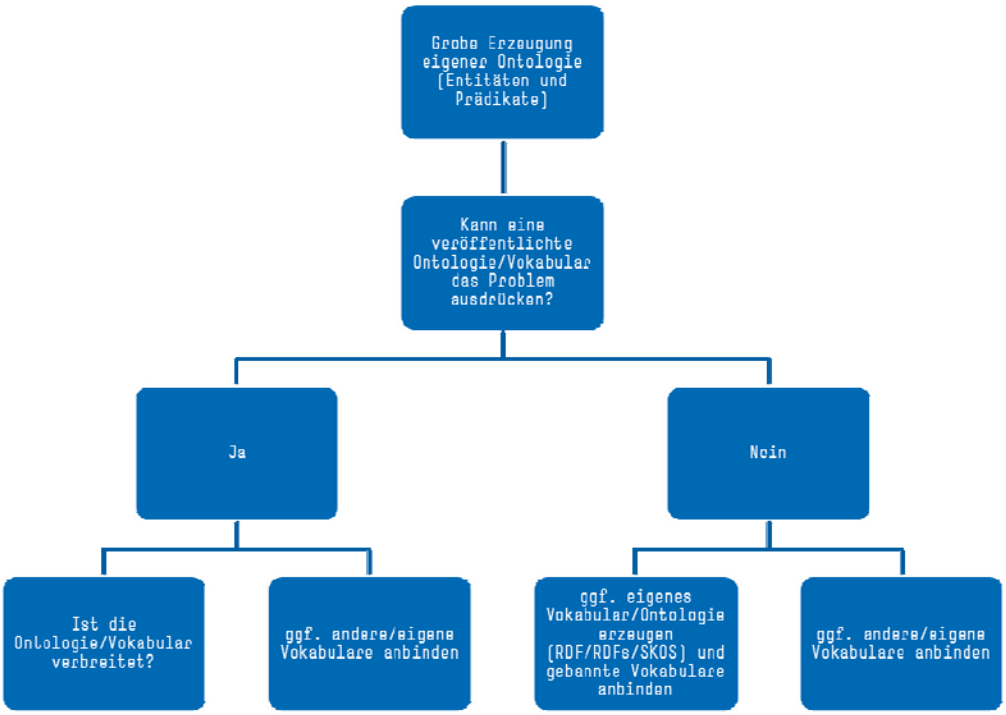


Abbildung 6-14: LOD-Workflow

Zunächst muss eine eigene Ontologie erstellt werden, in der alle Entitäten und deren Prädikate innerhalb (Attribute) und untereinander vorhanden sind, vgl. Kapitel 6.1.1 und Anhang F: Eigene Ontologie. Kann eine bereits veröffentlichte Ontologie bzw. Vokabular[324] all diese Aspekte abdecken, ist sie zur Verwendung geeignet. Ist die Ontologie weit verbreitet stellt dies ein weiteres Qualitätsmerkmal dar. Kann die Ontologie jedoch nur Teilbereiche abdecken, sollte man andere bzw. eigene Vokabulare verwenden, um eindeutige Repräsentationen zu erhalten. Kann keine bestehende Ontologie das Problem lösen, muss eine eigene (z.B. mit Hilfe von RDF/RDFs oder SKOS) entwickelt und bestenfalls mit bereits bestehenden Vokabularien verfeinert werden. Zu Erläuterung der Modellierungsformen werden der Potter Balbinus, das Fragment 30170 und der Findspot Langenhain herangezogen.

Kapitel 3 zeigt keine einfache konkrete Modellierung historischer Personen (inkl. Töpfer) in anderen Projekten. Lediglich moderne Personen können im Rahmen des FOAF-Projekts[325] abgebildet werden. Hier ist insbesondere der Aspekt der Beziehungen unter verschiedenen Personen (z.B. Person A kennt Person B) zu nennen, welches eine direkte Verbindung als Linked Data darstellt.

---

[323] siehe auch https://docs.google.com/document/d/1LqSsSchwlWhdn7S6IONOH3yju4-rn7cT6rX-gaqCU4c/edit?pli=1 (abgerufen 04.12.2013)

[324] siehe auch http://lov.okfn.org/dataset/lov (abgerufen 04.12.2013)

[325] vgl. http://www.foaf-project.org (abgerufen 04.12.2013)

Ein Potter kann jedoch auch als moderne Person interpretiert werden, welcher zwar nicht mehr physisch, aber in der heutigen Forschung weiterlebt. Bibliotheksstandards wie VIAF[326] beschreiben den Bestand von Personen mit FOAF[327] (vgl. Kapitel 3). Eine Verbindung zu diesen Datenbanken, wäre über eine eigene FOAF Repräsentation der Töpfer einfach möglich. Das FOAF Vokabular[328] gibt die Möglichkeit Personen bzw. Organisationen zu modellieren, der Beruf des Töpfers hingegen kann nicht abgebildet werden. Das Whois Vokabular[329] nutzt FOAF und erweitert es unter anderem um eine Möglichkeit der Angabe eines Berufs (vgl. Kapitel 3). Die zeitliche Einordnung des schon verstorbenen Töpfers, kann über die Angabe einer Zeitperiode (z.B. mit Pleiades Time Periods[330]) erfolgen. Abbildung 6-15 zeigt die Kombination der zuvor beschrieben Vokabularien und Konzepte Whois, FOAF und Pleiades Time Period zur Repräsentation des Potter Balbinus.

```
#Representation in WHOIS Syntax
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix whois: <http://www.kanzaki.com/ns/whois#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix time: <http://www.w3.org/2006/time#>.
@prefix pleiadestp: <http://pleiades.stoa.org/vocabularies/time-periods/>.
@prefix potter: <http://143.93.114.104/rest/samian/potters/>.
potter:Balbinus whois:career _:blanknode;
  a foaf:Person;
  foaf:name "Balbinus";
  foaf:gender "male";
  time:inside pleiadestp:roman.
_:blanknode a whois:Job;
rdfs:label "Potter".
```

Abbildung 6-15: Balbinus (WHOis) [331]

Die Modellierung der Findspots und Sites stellt eine größere Herausforderung dar. Eine Ontologie sollte die Beziehung zwischen diesen Entitäten eindeutig beschreiben. Das Pleiades Place Konzept[332] bietet solch ein Beziehungskonzept inklusive einiger anderer hilfreicher Attribute wie Ortstypen[333] und Zeitperioden[334] (vgl. Kapitel 3). Das Konzept ist somit primär zu verwenden. Weitere Konzepte wie CIDOC CRM[335] bzw. Geo-Vokabulare[336] beschreiben nicht in dieser Güte die vorhandene Problemstellung (vgl. Kapitel 3). Auffallende Unterschiede zwischen Samian Places und

---

[326] vgl. http://viaf.org (abgerufen 04.12.2013)

[327] Beispiel Julius Cäsar: http://viaf.org/viaf/100227925/rdf.xml (abgerufen 04.12.2013)

[328] vgl. http://xmlns.com/foaf/spec (abgerufen 04.12.2013)

[329] vgl. http://www.kanzaki.com/ns/whois# (abgerufen 04.12.2013)

[330] vgl. http://pleiades.stoa.org/vocabularies/time-periods (abgerufen 04.12.2013)

[331] vgl. http://143.93.114.104/rest/samian/potters/Balbinus.ttl (abgerufen 04.12.2013)

[332] vgl. http://pleiades.stoa.org/help/technical-intro-places (abgerufen 04.12.2013)

[333] vgl. http://pleiades.stoa.org/vocabularies/place-types (abgerufen 04.12.2013)

[334] vgl. http://pleiades.stoa.org/vocabularies/time-periods (abgerufen 04.12.2013)

[335] vgl. http://www.cidoc-crm.org (abgerufen 04.12.2013)

[336] vgl. http://lov.okfn.org/dataset/lov/details/vocabularySpace_Geography.html (abgerufen 04.12.2013)

Pleiades Places werden in Kapitel 7.1. erläutert. Insbesondere die Tatsache, dass Namen nur Pleiades Places zugeordnet werden können, jedoch keiner Location, bedingt eine Modifikation des Konzepts, hier mit Hilfe von geonames:name[337]. Abbildung 6-16 zeigt die RDF/Turtle Repräsentation des Findspot Langenhain[338] nach Pleiades Syntax unter Angabe des Feature Type (settlement+findspot; wenn isKilnsite auch production), dem Namen nach Geonames, seiner Zeitperiode, der Position und seinen anhängenden Locations. Locations sind nach dem gleichen Schema modelliert, vgl. Langenhain__store[339].

```
#Representation like Pleiades Syntax
@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix geovocab: <http://geovocab.org/spatial#> .
@prefix osgeo: <http://data.ordnancesurvey.co.uk/ontology/geometry/> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix gn: <http://www.geonames.org/ontology#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix pleiades: <http://pleiades.stoa.org/places/vocab#> .
@prefix pleiadespt: <http://pleiades.stoa.org/vocabularies/place-types/> .
@prefix pleiadestime: <http://pleiades.stoa.org/vocabularies/time-periods/> .
@prefix findspot: <http://143.93.114.104/rest/samian/findspots/> .
findspot:Langenhain a geovocab:Feature;
  rdfs:label "Langenhain";
  rdfs:comment "An ancient place";
  foaf:primaryTopicOf findspot:Langenhain.
findspot:Langenhain a pleiades:Place,
  skos:Concept;
  pleiades:hasFeatureType pleiadespt:settlement;
  pleiades:hasFeatureType pleiadespt:findspot;
  pleiades:hasLocation findspot:Langenhain__store;
  gn:name "Langenhain";
  geo:lat 50.365103;
  geo:long 8.644794;
  osgeo:asGeoJSON "{\"type\": \"Point\", \"coordinates\": [8.644794, 50.365103]}";
  osgeo:asWKT "POINT (8.644794 50.365103)";
  time:during pleiadestime:roman.
```

Abbildung 6-16: Langenhain (Pleiades)

Die semantische Modellierung von archäologischen Fragmenten und Objekten wird in anderen Projekten fast ausschließlich mit CIDOC CRM[340] gelöst (vgl. Kapitel 3). Auf Grund mangelnder Alternativen werden die Scherbenfragmente als E22: Man Made Object gesehen. Eine Modellierung nach CIDOC CRM ist komplex und mit erheblichem Aufwand verbunden und wurde nur in Ansätzen in dieser Arbeit durchgeführt. Den Scherbenfragmenten werden Typ, Status und weitere Merkmale zugeordnet. Die Modellierung ist jedoch nur als grober Entwurf zu sehen. Hier ist in

[337] vgl. http://www.geonames.org/ontology/documentation.html (abgerufen 04.12.2013)

[338] vgl. http://143.93.114.104/rest/samian/findspots/Langenhain.ttl (abgerufen 04.12.2013)

[339] vgl. http://143.93.114.104/rest/samian/findspots/Langenhain__store.ttl (abgerufen 04.12.2013)

[340] vgl. http://www.cidoc-crm.org (abgerufen 04.12.2013)

Zukunft noch Entwicklungsbedarf nötig. Abbildung 6-17 zeigt einen ersten Versuch der Modellierung nach CIDOC CRM des Fragments 30170[341].

```
#Representation like CIDOC CRM (Claros) Syntax
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix fragment: <http://143.93.114.104/rest/samian/fragments/>.
@prefix crm: <http://purl.org/NET/crm-owl#> .
@prefix arachne: <http://arachne.uni-koeln.de/vocabulary/objectType#>.
@prefix claros: <http://id.clarosnet.org/vocab/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
fragment:30170 a crm:E22_Man-Made_Object;
  rdfs:label "Fragment einer Terra Sigillata Scherbe mit Die eines Toepfers";
  crm:P102_has_title "Fragment einer Terra Sigillata Scherbe mit Die eines Toepfers";
  crm:P2_has_type "Samian Terra Sigillata";
  crm:P2_has_type "doubtful";
  crm:P57_has_number_of_parts "1";
  crm:P56_bears_feature "1a";
  crm:P44_has_condition [ a crm:E3_Condition_State;
    crm:P2_has_type "fragment"];
  crm:P45_consists_of "Ton".
```

Abbildung 6-17: Fragment 30170 (CIDOC CRM)

Die zuvor beschrieben Konzepte dienen zur Repräsentation der Attribute der Entitäten potter, fragment und findspot. Die Modellierung der Beziehungen untereinander (z.B. wurde gestempelt von, hat gearbeitet in, wurde hergestellt von, hat hergestellt, hat produziert) steht jedoch noch aus. Diese Beziehung könnte durch ein bestehendes Vokabular abgedeckt werden, falls vorhanden. Die Beziehungen untereinander sind jedoch nicht exakt definiert: ein Töpfer kann beispielsweise ein Fragment erstellt, gestempelt oder nur in Auftrag gegeben haben, bzw. ein Arbeiter seiner Töpferei dieses erstellt haben. Um diesen Umstand Rechnung zu tragen, ist ein Verlinkungskonzept zu bevorzugen, welches einen Interpretationsspielraum bietet und nicht an exakten Prädikaten festhält. Ein Konzept, das lediglich irgendeine Art von Beziehung einer Ressource mit einer anderen ausdrückt, ist Open Annotation (vgl. Kapitel 3). Lose Annotationen werden zur Beziehungsbeschreibung zwischen Töpfer und Fragment, Fragment und Findspot, sowie Findspot und Töpfer genutzt. Hierbei dient die eigene Ressource immer als target, die damit verbundene Ressource als Body, vgl. Beziehungen des Fragments 30170 in Abbildung 6-18. Zur Repräsentation eindeutiger Beziehungen ist ein eigenes Vokabular notwendig.

[341] vgl. http://143.93.114.104/rest/samian/fragments/30170.ttl

```
#Links to Findspots and Potters
@prefix oa: <http://www.openannotation.or4g/ns/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix anno: <http://143.93.114.104/rest/samian/annotations#>.
@prefix agent: <http://143.93.114.104/rest/samian/agents#>.
@prefix potter: <http://143.93.114.104/rest/samian/potters/>.
@prefix findspot: <http://143.93.114.104/rest/samian/findspots/>.
@prefix fragment: <http://143.93.114.104/rest/samian/fragments/>.
anno:30170-Langenhain a oa:Annotation;
  oa:motivatedBay oa:linking;
  oa:annotatedBy agent:thiery;
  oa:annotatedAt "Wed Nov 20 14:44:20 CET 2013";
  oa:hasBody findspot:Langenhain;
  oa:hasTarget fragment:30170;
  dcterms:description "Link made by Heinz/Mees/Thiery. Database by RGZM. Potter has worked in Findspot or exported to it.".
anno:30170-Balbinus a oa:Annotation;
  oa:motivatedBay oa:linking;
  oa:annotatedBy agent:thiery;
  oa:annotatedAt "Wed Nov 20 14:44:20 CET 2013";
  oa:hasBody potter:Balbinus;
  oa:hasTarget fragment:30170;
  dcterms:description "Link made by Heinz/Mees/Thiery. Database by RGZM. Potter has stamped the fragment.".
agent:thiery a foaf:person;
  foaf:name "Florian Thiery".
```

Abbildung 6-18: Annotation zwischen Fragment 30170 und anderen Entitäten

6.1.6 Pelagios Entry-Point

Analog zu den Pelagios Datasets[342] werden die Annotation, welche das Mapping zu den Pleiades IDs beschreiben, in einem Entry-Point zur Verfügung gestellt, vgl. CLAROS[343], American Numismatic Society[344] und Regnum Francorum Online[345]. Alle vorliegenden Verknüpfungen zwischen Findspots und deren Entsprechungen im Datenbestand von Pleiades sind in Turtle- Syntax über den Entry-Point `http://143.93.114.104/rest/samian/pleiades` erreichbar (vgl. Abbildung 6-19). Das Verfahren des manuellen Mapping, sowie die RDF-Techniken zur Verknüpfung der Datenbestände, werden in Kapitel 7 erläutert.

[342] vgl. http://pelagios.dme.ait.ac.at/api/datasets (abgerufen 04.12.2013)

[343] vgl. http://data.clarosnet.org/graph/pelagios (abgerufen 04.12.2013)

[344] vgl. http://numismatics.org/pelagios.rdf (abgerufen 04.12.2013)

[345] vgl. http://www.francia.ahlfeldt.se/api/export_pelagios_n3.php (abgerufen 04.12.2013)

```
#Links to Pleiades
#Prefix
@prefix oa: <http://www.openannotation.or4g/ns/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix pleiadesplace: <http://pleiades.stoa.org/places/> .
@prefix anno: <http://143.93.114.104/rest/samian/annotations#>.
@prefix agent: <http://143.93.114.104/rest/samian/agents#>.
@prefix findspot: <http://143.93.114.104/rest/samian/findspots/>.

anno:Baldock-79311 a oa:Annotation;
  oa:motivatedBy oa:linking;
  oa:annotatedBy agent:Mees;
  oa:annotatedAt "Tue Nov 12 17:34:54 CET 2013";
  oa:hasBody pleiadesplace:79311;
  oa:hasTarget findspot:Baldock;
  dcterms:description "Link made by Heinz/Mees/Thiery.".
agent:Mees a foaf:person;
  foaf:name "Allard Mees".
anno:Ribchester-79352 a oa:Annotation;
  oa:motivatedBy oa:linking;
  oa:annotatedBy agent:Mees;
  oa:annotatedAt "Tue Nov 12 17:34:54 CET 2013";
  oa:hasBody pleiadesplace:79352;
  oa:hasTarget findspot:Ribchester;
  dcterms:description "Link made by Heinz/Mees/Thiery.".
agent:Mees a foaf:person;
  foaf:name "Allard Mees".
```

Abbildung 6-19: Pelagios Entry-Point

Strukturelle Metadaten können dem Mapping-Datensatz mit Hilfe des VoID[346] Vokabulars[347] hinzugefügt werden. Die Grundidee von VoID ist die Verfügbarkeit einer zusätzlichen begleitenden RDF-Datei, welche eine Hierarchie von datasets und subsets definiert. Des Weiteren kann die Datei beschreibende Metadaten wie Erläuterungen, Labels und Lizenzinformationen aufführen. Das Pelagios-Projekt nutzt ebenfalls einige voID Funktionalitäten [GITHUB, 2013]. Beispiele[348] für die Beschreibung der Links der eigenen Ressourcen zu Pleiades IDs sind u. A. bei Regnum Francorum Online[349], Open Context[350] und Nomisma[351] zu finden. Neben dem Pelagios Entry-Point verfügt die ReST-Schnittstelle auch über ein voID-File (verfügbar in Turtle-Syntax[352]) zur Beschreibung des eigenen Verlinkungs-Datensatzes, welche über die URI

http://143.93.114.104/rest/samian/void.ttl

zu erreichen ist (vgl. Abbildung 6-20).

---

[346] vgl. http://www.w3.org/TR/void (abgerufen 04.12.2013)

[347] vgl. http://vocab.deri.ie/void (abgerufen 04.12.2013)

[348] vgl. https://github.com/pelagios/pelagios-monitor/blob/master/predefined_datasets.yaml (04.12.2013)

[349] vgl. http://www.francia.ahlfeldt.se/downloads/rfo_void.rdf (abgerufen 04.12.2013)

[350] vgl. https://github.com/ekansa/open-context-code/blob/master/application/views/scripts/export/pelagios-void.phtml (abgerufen 04.12.2013)

[351] vgl. https://github.com/ewg118/nomisma/blob/master/cocoon/xml/pelagios.void.rdf (04.12.2013)

[352] vgl. http://www.w3.org/ns/formats (abgerufen 04.12.2013)

```
@prefix : <http://143.93.114.104/rest/samian/pleiades> .
@prefix void: <http://rdfs.org/ns/void#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
:oc a void:Dataset;
# Title and description
dcterms:title "Sinked Samian Ware Data";
dcterms:description "Potter, Pots and Places of Samian Ware";
# License
dcterms:license <http://creativecommons.org/licenses/by-nc-nd/3.0/deed.de>;
# Dump file location and serialization format
void:dataDump <http://143.93.114.104/rest/samian/pleiades>;
void:feature <http://www.w3.org/ns/formats/Turtle>;
# A homepage with information ABOUT the data
foaf:homepage <http://www.rgzm.de/samian/>;
```

Abbildung 6-20: VoID-File

6.1.7 Bereitstellungen für den Explorer

Der in Kapitel 8.2 beschriebene Linked Samian Ware Explorer (LSWE) nutzt zwei speziell zugeschnittene Datenformate zur Visualisierung der Linked Data, welche mit Hilfe der ReST-Schnittstelle realisiert werden: Findspots eines Potters, sowie die Potter (und deren Fragmente) und Fragmente (und deren Potter) eines Findspot. Orte, die mit einem Töpfer in Verbindung stehen, können über das nachfolgende Interface als GeoJSON abgerufen und direkt in eine Web-Kartenanwendung wie z.B. Leaflet integriert werden:

- http://143.93.114.104/rest/samian/potters/{potter}.exp

Zur Visualisierung der Links zwischen einem Findspot und dessen Potter und Fragments wird ein XML-Interface verwendet, das mit JavaScript leicht zu parsen und anzuzeigen ist:

- http://143.93.114.104/rest/samian/findspots/{findspot}.exp

Abbildung 6-21 (links) zeigt exemplarisch die Findspots des Potters Balbinus[353], rechts ist ein Ausschnitt, der mit dem Findspot Langenhain[354] verknüpften Potter und Fragments zu sehen. Details zur Umsetzung in JAVA werden in Kapitel 6.1.9 erläutert, Informationen zur Funktionsweite des Explorers und die Einbindung der *.exp sind Kapitel 8.2 zu entnehmen.

[353] vgl. http://143.93.114.104/rest/samian/potters/Balbinus.exp (abgerufen 04.12.2013)

[354] vgl. http://143.93.114.104/rest/samian/findspots/Langenhain.exp (abgerufen 04.12.2013)

```
{
"type": "FeatureCollection",
"features": [
{"type": "Feature",
"properties": {
"kilnsite": "false",
"name": "Caerleon"
},
"geometry": {
"type": "Point",
"coordinates": [-2.950000048, 51.61666107]
}
},
{"type": "Feature",
"properties": {
"kilnsite": "false",
"name": "Caistor-by-Norwich"
},
"geometry": {
"type": "Point",
"coordinates": [1.291995287, 52.58224106]
}
},
{"type": "Feature",
"properties": {
"kilnsite": "false",
"name": "Carlisle"
},
"geometry": {
"type": "Point",
"coordinates": [-2.93333292, 54.88333893]
}
},
{"type": "Feature",
"properties": {
"kilnsite": "false",
"name": "Carmarthen"
},
"geometry": {
"type": "Point",
"coordinates": [-4.311666489, 51.85916519]
}
},
{"type": "Feature",
"properties": {
"kilnsite": "false",
"name": "Cirencester"
},
"geometry": {
"type": "Point",
"coordinates": [-1.983332992, 51.73332977]
}
},
```

```
- <list>
  - <potter uri="http://143.93.114.104/rest/samian/potters/Petrullus">
    - <fragment_p>
        <uri>http://143.93.114.104/rest/samian/fragments/111212</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111216</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111218</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111220</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111224</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111228</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111236</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111237</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111240</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111241</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111244</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111247</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111250</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111254</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111259</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111264</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111267</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111304</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111306</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111310</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111311</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111313</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111317</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111322</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111326</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111332</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111337</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111345</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111346</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111347</uri>
        <uri>http://143.93.114.104/rest/samian/fragments/111351</uri>
    </fragment_p>
  </potter>
  + <potter uri="http://143.93.114.104/rest/samian/potters/Balbinus"></potter>
  + <potter uri="http://143.93.114.104/rest/samian/potters/Attianus_ii"></potter>
  + <potter uri="http://143.93.114.104/rest/samian/potters/Criciro_vii"></potter>
  + <potter uri="http://143.93.114.104/rest/samian/potters/Aper_iii"></potter>
  - <potter uri="http://143.93.114.104/rest/samian/potters/Felix_ii">
```


Abbildung 6-21: Explorer-Bereitstellungen

6.1.8 Triple-Dateien zum Befüllen des Triplestore

Die RDF-Repräsentationen der Findspots, Fragments und Potter und deren Verlinkungen werden für jede Ressource aus der Datenbank neu erstellt. Eine direkte Verbindung zum Sesame Triplestore (vgl. Kapitel 6.3) besteht nicht. Zum Export der RDF-Darstellungen und der Annotationen werden aufgrund der großen Datenmenge neue Dateien angelegt, welche unter den folgenden URIs in Turtle Syntax verfügbar sind:

- http://143.93.114.104/share/potters.ttl
- http://143.93.114.104/share/fragments.ttl
- http://143.93.114.104/share/findspots.ttl

Die Dateien können mit Hilfe der nachfolgenden Links erstellt werden (Details siehe Kapitel 6.1.9):

- http://143.93.114.104/rest/samian/potters.ttl (abgerufen 04.12.2013)
- http://143.93.114.104/rest/samian/fragments.ttl (abgerufen 04.12.2013)
- http://143.93.114.104/rest/samian/findspots.ttl (abgerufen 04.12.2013)

Die Erstellung der Dateien und aller Verlinkungen (mehr als eine Million Triples) kann mehrere Stunden in Anspruch nehmen. Der Prozess kann mittels Konsole überwacht und ggf. angehalten

werden. Nach dem erstmaligen Erstellen der Dateien sind aus Gründen der Performance nur noch diese zu verwenden. Ein direkter Zugriff auf die create-URL ist zu vermeiden. Die mit diesem Verfahren erstellten Triple können nun in den Triplestore überführt werden (vgl. Kapitel 6.3).

6.1.9 Programmiertechnische Umsetzung

JAVA bietet als plattformunabhängige und objektorientierte Programmiersprache[355] vielerlei Möglichkeiten der Nutzung auf dem GeInArFa-Server. Erweiterbarkeit und vor allem die Unterstützung performanter und von der Community genutzter Bibliotheken und Frameworks (auch für Geo-Anwendungen, z.B. GeoTools[356]) und die Einbindung dieser, inklusive aller Abhängigkeiten (hier mit Maven[357]), bieten schnelle und robuste Lösungen. Des Weiteren nutzen Geoserver[358] (vgl. Kapitel 2.3) und Sesame[359] (vgl. Kapitel 2.5.5) JAVA zur programmiertechnischen Umsetzung der Tools, woraus sich JAVA als Programmiersprache und -Umgebung anbietet. Die Daten liegen in einer PostGIS Datenbank vor (vgl. Kapitel 5). Es ist somit nach einer Lösung zu suchen, welche die Unterstützung eines ReSTful Web Services (vgl. Kapitel 2.4), die Bereitstellung von XML/JSON wie auch Linked Data (RDF) Elementen sicherstellt und zudem den Geoserver, sowie die PostGIS Datenbank integriert. Server Systeme wie Virtuoso[360] bieten sowohl XML und RDF-Unterstützung, stellen einen Linked Data und Web Application Server zur Verfügung und können mit Hilfe von GeoSPARQL[361] räumliche Abfragen tätigen[362]. Sie bieten weiterhin eine direkte Integration von relationalen Datenbanken[363] (z.B. PostGIS) an. Die Einbindung eines Geoservers ist jedoch nicht vorgesehen. Aufgrund der komplexen Struktur der Virtuoso-Server-Struktur wird im Rahmen dieser Masterarbeit auf die Verwendung verzichtet. Zukünftige Projekte sollten jedoch auf den Server zurückgreifen.

Die Datenbereitstellung der ReST-Schnittstelle ist als Server Programm (rest.war) im Apache Tomcat verfügbar und basiert auf folgenden Frameworks, welche mit Hilfe von Maven eingebunden werden können: Jersey[364], JAXB[365] und Hibernate Spatial[366]. Jersey ist ein Open Source Fra-

---

[355] vgl. http://openbook.galileocomputing.de/javainsel/javainsel_01_002.html#dodtpaef1ef41-da01-4368-8372-4f815650cf09 (abgerufen 04.12.2013)

[356] vgl. http://www.geotools.org (abgerufen 04.12.2013)

[357] vgl. http://maven.apache.org (abgerufen 04.12.2013)

[358] vgl. http://geoserver.org/display/GEOS/Welcome (abgerufen 04.12.2013)

[359] vgl. http://www.openrdf.org/index.jsp (abgerufen 04.12.2013)

[360] vgl. http://virtuoso.openlinksw.com (abgerufen 04.12.2013)

[361] vgl. http://www.opengeospatial.org/standards/geosparql (abgerufen 04.12.2013)

[362] vgl. http://svn.aksw.org/projects/GeoKnow/Public/D2.1.1_Market_and_Research_Overview.pdf (abgerufen 04.12.2013)

[363] vgl. http://virtuoso.openlinksw.com/images/varch625.jpg (abgerufen 04.12.2013)

[364] vgl. http://mvnrepository.com/artifact/com.sun.jersey (abgerufen 04.12.2013)

[365] vgl. https://jaxb.java.net (abgerufen 04.12.2013)

mework und bietet Entwicklern einen RESTful Webservice an, welcher die Offenlegung aller gängigen Medientypen ermöglicht [JERSEY, 2013]. Jersey unterstützt alle gängigen Methoden wie HTTP GET, POST, DELETE und UPDATE und lässt einen Response an den Client, sowohl als einfachen String, wie auch als JAVA Objekt[367],[368] zu. JAXB ermöglicht die Erstellung von XML-Objekten[369] (als JAVA-Klassenstrukturen) und somit einen möglichen Response jener mit Hilfe von Jersey[370] [VOGEL, 2012]. Hibernate Spatial (HS) ist eine generische Open Source-Erweiterung des Hibernate[371] Frameworks zur Handhabung geografischer Daten und nutzt die meisten Funktionen der OGC Simple Feature Specification[372]. HS ermöglicht den Umgang mit geografischen Daten in standardisierten Formaten und bietet eine Cross-Datenbank-Schnittstelle zur Datenspeicherung und -abfrage [HIBERNATESPATIAL, 2013]. Die verwendete PostGIS Datenbank bietet Möglichkeiten der Objektorientierung[373]. Eine einfache SELECT-Abfrage durch ein JAVA Programm erzwingt keine Rückgabe als Objekt. Dieses muss zur weiteren Befragung in JAVA in ein Objekt umgewandelt werden. Dieser Vorgang wird als Object-Relational-Mapping bezeichnet und von Hibernate geleistet. Das Speichern der Daten wird als Persistenz bezeichnet. Hibernate bietet so genannte Persitenz-Layer an, welche das Abfragen, Bearbeiten und Löschen von Daten ermöglicht[374] und trennt somit die Anwendungsschicht von der Datenbank. In dieser Arbeit wird sich auf das Lesen von Datenbankinformationen beschränkt [HENNEBRÜDER, 2007]. Insbesondere die Erweiterbarkeit der Anwendung, die einfache Austauschbarkeit des Backends (Datenbanktyp) und der Performancevorteil beim Laden großer Datenmengen durch Lazy Loading[375] spricht für die Nutzung von Hibernate, wird im Programm dbutils.jar realisiert und als Dependency eingebunden.

---

[366] vgl. http://www.hibernatespatial.org (abgerufen 04.12.2013)

[367] siehe auch Tutorial: http://www.vogella.com/articles/REST (abgerufen 04.12.2013)

[368] siehe auch Tutorial: http://www.ibm.com/developerworks/library/wa-aj-tomcat (abgerufen 04.12.2013)

[369] vgl. http://www.vogella.com/articles/JAXB/article.html (abgerufen 04.12.2013)

[370] vgl. http://www.mkyong.com/webservices/jax-rs/download-xml-with-jersey-jaxb (abgerufen 04.12.2013)

[371] vgl. http://www.hibernate.org (abgerufen 04.12.2013)

[372] vgl. http://www.opengeospatial.org/standards/sfa (abgerufen 04.12.2013)

[373] vgl. http://www.postgresql.org/docs/6.3/static/c0101.htm (abgerufen 04.12.2013)

[374] vgl. http://www.hibernatespatial.org/tutorial-hs4.html (abgerufen 04.12.2013)

[375] vgl. http://java.dzone.com/articles/lazyeager-loading-using (abgerufen 04.12.2013)

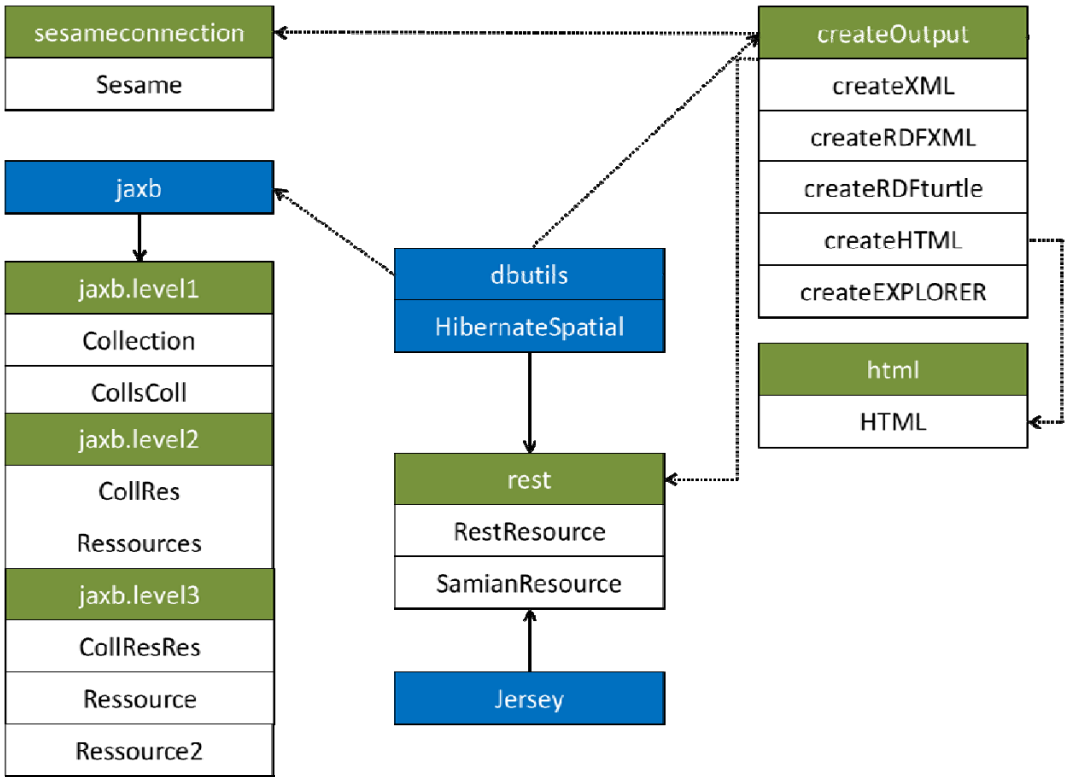


Abbildung 6-22: Klassenstruktur der ReST-Anwendung

Abbildung 6-22 zeigt die Klassenstruktur der JAVA ReST-Anwendung und deren Zusammenspiel mit den zuvor vorgestellten Bibliotheken Jersey, JAXB und Hibernate Spatial. Als ansprechbares Servlet dienen die mit Jersey verbunden Klassen, Rest- und SamianResource. Sie verwalten den ReST-konformen In- und Output der Daten einer relativen URI mit den Annotationen @GET[376] und @Path[377]. Die Annotation @Produces[378] steuert den Output der Daten in gängigen Formaten (text/plain, application/xml, application/json). Eine HTTP GET Anfrage kann Parameter enthalten, welche über die Annotation @Consumes[379] gelesen werden können (dies wird jedoch nicht angewendet). Die Klassen der Jersey-ReST-Ressourcen erzeugen je nach Anfrage eine Instanz der createOutput-Klassen und somit ein JAVA-Objekt oder einen Outputstring, welches in standardisierten Formaten an den Client gesendet wird.

XML/JSON-Anfragen der ersten drei Ebenen der ReST-Schnittstelle (/rest, /rest/samian, /rest/samian/{ressources}) werden mit Hilfe von JAXB und somit mit der Erzeugung von Klassenstrukturen und Objekten (ggf. mit Anfragen an die Datenbank→Hibernate) realisiert. Dies bietet den Vorteil, dass jene Objekte, je nach Anfrage im Request Header, ohne manuelle Bearbeitung als XML- oder JSON-Struktur an den Client weitergegeben werden können. Eine Repräsentation als HTML wird durch kein Framework geleistet. Die Klassen CreateHTML und HTML ermöglichen

[376] vgl. https://jersey.java.net/documentation/latest/user-guide.html#d0e1290 (abgerufen 04.12.2013)

[377] vgl. https://jersey.java.net/documentation/latest/user-guide.html#d0e1240 (abgerufen 04.12.2013)

[378] vgl. https://jersey.java.net/documentation/latest/user-guide.html#d0e1328 (abgerufen 04.12.2013)

[379] vgl. https://jersey.java.net/documentation/latest/user-guide.html#d0e1407 (abgerufen 04.12.2013)

dies jedoch. XML/JSON- und RDF-Ergebnisse von Anfragen konkreter Ressourcen (/rest/samian/{ressources}/{ressource}) werden nicht mit Hilfe von JAXB in eine Klassenstruktur überführt, da dies (z.B. MIDAS XML) eine sehr tiefe Modellierung voraussetzt, welche für diesen Prototypen nicht erforderlich ist. In zukünftigen Projekten jedoch ist auch hier die Modellierung in JAXB (für XML und JSON) und eine Modellierung mit anderen Bibliotheken für RDF-Repräsentationen vorzusehen. Die Ergebnisse werden für konkrete Ressourcen in einen langen String (nach XML/JSON/RDF-Spezifikation) überführt und an den Client zurückgesendet. Ein String, welcher XML-Elemente enthält und mit dem MIME Type[380] application/xml an den Client gesendet wird, wird jedoch auch als XML-Response verstanden.

Die für die Bereitstellung von HTML-, XML-, JSON- oder RDF-Repräsentation benötigten Daten werden durch das Programm dbutils.jar und Hibernate Spatial zur Verfügung gestellt. Mit der Verwendung der Java Persistence API (JPA) sind Entities als einfache POJOs (Plain Old Java Object) verfügbar, werden objektorientierte Klassenhierarchien mit Vererbung, Assoziationen, Polymophismus, etc. unterstützt und eine objektinterne Abfragesprache (JPQL, ähnlich zu SQL) verwendbar [RÖDER, 2010]. Ein Entity Manger übernimmt die Transaktionen zur Datenbank und bildet somit das Herzstück der Anwendung. Die Klasse EventManger steuert gezielte Anfragen (z.B. nach einer Liste von Töpfern oder nach einem bestimmten Töpfer-Objekts) mit Hilfe des Entity Mangers (z.B. Select p from Potter p oder Select p from Potter p where p.name={name}) und übergibt diese Objekte an die ReST-Anwendung. Jede Datenbanktabelle ist als persistente POJO Klasse mit allen Attributen mit Hilfe einer @Entity[381] Annotation in JAVA modelliert und stellt get- und set-Methoden bereit. Die Tabelle Fragment bietet eine Besonderheit. Sie besitzt Beziehungen zu den Tabellen Potter und Findspot. Diesem Umstand kann mit der Annotation @ManyToOne[382] Rechnung getragen werden. Der Fremdschlüssel wird mit @JoinColumn[383] verknüpft. Viele Datenbankzugriffe können ein Grund für Performanceprobleme sein. Je nach Anwendungsfall muss daher eine geeignete Fetching-Strategie gewählt werden. JPA definiert zwei Strategien: Lazy Load oder Eager Load. Mit einem Lazy Loading werden die Daten erst geladen, wenn ein Zugriff auf das Attribut oder Objekt erfolgt. Ein sofortiges vollständiges Laden aller Daten kann mit Eager Load realisiert werden [RÖDER, 2010]. Aufgrund der großen Datenmenge an Fragments wird in dbultils ein Lazy Loading durchgeführt. Dieses Verfahren jedoch birgt ein Problem in sich: Wird der Entity Manager vor dem Nachladen der Potter und Findspot Entitys geschlossen (was der Fall ist) führt dies zu einer Exception [RÖDER, 2010]. In diesem Fall werden die Objekte bei einer einmaligen Transaktion geladen und als String zur Weiterverarbeitung zwischengespeichert.

Auf Grund der Komplexität der ReST-Anwendung wird an dieser Stelle auf ein ausführliches Beispiel verzichtet. In diesem Fall sei auf den Quellcode verwiesen.

---

[380] vgl. http://de.wikipedia.org/wiki/Internet_Media_Type (abgerufen 04.12.2013)

[381] vgl. http://docs.jboss.org/hibernate/annotations/3.5/reference/en/html/entity.html (04.12.2013)

[382] vgl. auch http://www.dzone.com/tutorials/java/hibernate/hibernate-example/hibernate-mapping-many-to-one-using-annotations-1.html (abgerufen 04.12.2013)

[383] vgl. http://www.galileocomputing.de/artikel/gp/artikelID-328/position-2 (abgerufen 04.12.2013)

Ein besonderes Merkmal weisen die Kapitel 6.1.7 gezeigten Bereitstellungen für den Linked Samian Ware Explorer auf. Wohingegen zur Erstellung der gezeigten Repräsentationen auf die PostGIS Datenbank via Hibernate oder den Geoserver zugegriffen wird, werden für die Anfragen des Explorers, die in den Triplestore übertragenen Daten, genutzt. Die Verbindung mit dem Triplestore wird mit der Klasse Sesame realisiert. Die Erstellung des Outputs übernimmt die Klasse createEXPLORER. Kapitel 6.1.7 verweist auf zwei unterschiedliche Repräsentationen: ein GeoJSON aller Findspots eines Töpfers, sowie eine XML-Struktur, in der alle Verbindungen von Fragmenten zu einem Töpfer und die Verbindung eines Töpfers zu einem Fragment an einem bestimmten Findspot dargestellt sind. Eine Abfrage des Triplestores kann mit den, in den Kapiteln 6.3 und 7.4 vorgestellten Methoden und der query URL des Triplestore, erfolgen.

Eine SPARQL Abfrage an den Triplestore (siehe unten mit Beispiel Balbinus) ergibt eine Liste an Findspots, die mit einem Töpfer in Verbindung gebracht werden. Nach dem Parsen der Antwort XML und der Untersuchung des Place Types (ist das Objekt pleiadespt:production enthalten?), wird das geoJSON Objekt (hier als langer String) erzeugt und als application/json an den Client gesendet:

```
SELECT DISTINCT ?erg ?lat ?lon ?name ?ft WHERE {
    ?erg pleiades:hasFeatureType ?ft .
    ?erg gn:name ?name .
    ?erg geo:lat ?lat .
    ?erg geo:long ?lon .
    { ?erg rdf:type pleiades:Place . } UNION
    { ?erg rdf:type pleiades:Location . }
    ?a oa:hasBody ?erg .
    ?a oa:hasTarget potter:Balbinus .
}
```

Die Erstellung des Potter/Fragment-XML eines Findspot folgen ebenfalls einem ähnlichen Schema: Abfrage des Triplestore, Parsen der Antwort XML und Erzeugung einer XML Struktur, welche als application/xml ebenfalls an den Client gesendet wird. Hierbei sind mehrere zwei SPARQL Abfragen nötig (Beispiele. Balbinus, Langenhain und Fragment 30170):

1. Alle Potter eines Findspots

```
SELECT DISTINCT ?erg WHERE {
    ?erg rdf:type foaf:Person .
    ?a oa:hasBody ?erg .
    ?a oa:hasTarget findspot:Langenhain .
}
```

2. Alle Fragments eines Potters

```
SELECT DISTINCT ?erg WHERE {
    ?erg rdf:type crm:E22_Man-Made_Object .
    ?a oa:hasBody ?erg .
    ?a oa:hasTarget potter:Balbinus .
}
```

3. Alle Fragments eines Findspot

```
SELECT DISTINCT ?erg WHERE {
    ?erg rdf:type crm:E22_Man-Made_Object .
    ?a oa:hasBody ?erg .
    ?a oa:hasTarget findspot:Langenhain .
}
```

4. Töpfer eines Fragments

```
SELECT DISTINCT ?erg WHERE {
    ?erg rdf:type foaf:Person .
    ?a oa:hasBody ?erg .
    ?a oa:hasTarget fragment:30170 .
}
```

6.2 OGC-Webdienste

Die in der PostGIS-Datenbank vorliegenden Geometrien der Findspots (vgl. Kapitel 5) und die in der ReST-Schnittstelle (vgl. Kapitel 6.1) vom Geoserver daraus erzeugten GML- bzw. GeoJSON-Repräsentation, bilden bereits eine erste Anwendung zur Nutzung von OGC Webdiensten. Um auch andere Anwendungen, wie z.B. Leaflet[384], mit Daten zu versorgen, ist eine Bereitstellung von Web Map Services (WMS) und Web Feature Services (WFS) vorstellbar (vgl. Kapitel 2.1). Im Rahmen dieser Masterarbeit werden die zuvor beschriebenen Services mit einem Geoserver[385] (vgl. Kapitel 2.3) bereitgestellt.

Die bereitzustellenden Daten benötigen einen eindeutigen Bezeichner (Name des Arbeitsbereichs und Namensraum URI), hier 'Samian' (vgl. Abbildung 6-23, oben). Dem Arbeitsbereich können verschiedene Datenspeicher zugeordnet werden, welcher in diesem Fall mit der schon bestehenden PostGIS Datenbank und dessen Schema public (vgl. Kapitel 5) realisiert wird (vgl. Abbildung 6-23, mitte) und dabei die Services WMS und WFS freigeschaltet werden. Aus den verbundenen Datenspeichern ist nun die Bereitstellung von einzelnen Layern möglich. Hierzu werden die in der Datenbank erstellte Tabelle findspot, sowie aus jener generierte Views (vgl. Kapitel 5.3) genutzt und sind als EPSG:4326 Geometrie im Namensraum Samian verfügbar (vgl. Abbildung 6-23, unten).

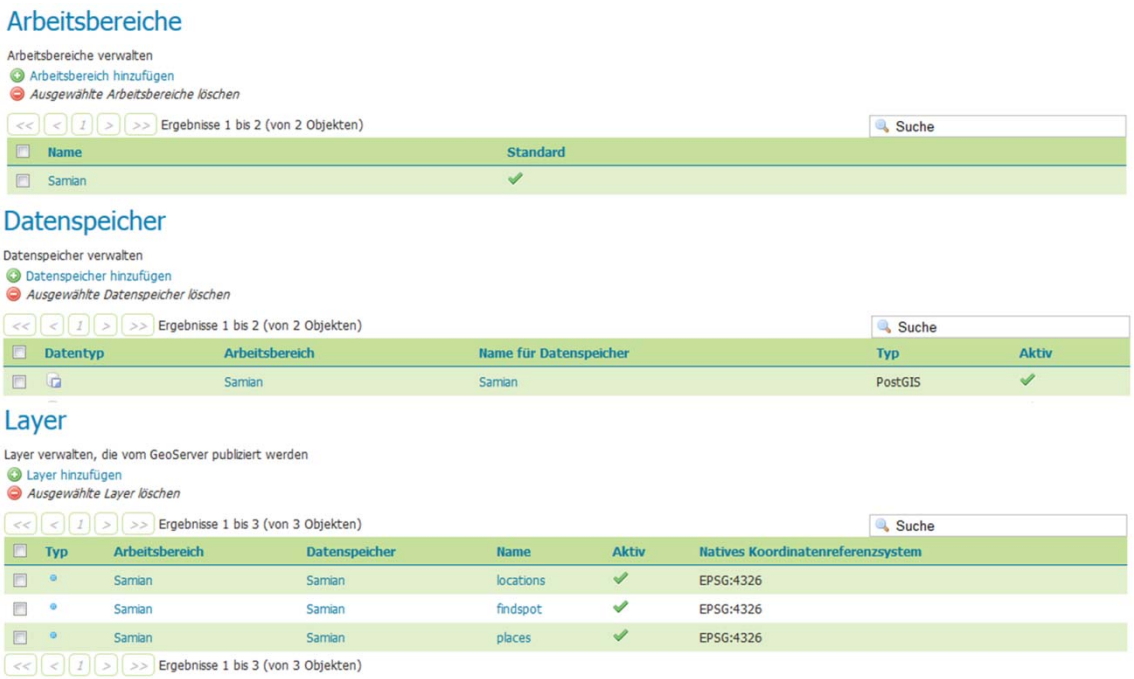


Abbildung 6-23: Geoserver (Struktur)

[384] vgl. http://leafletjs.com (abgerufen 04.12.2013)

[385] URL des genutzten Geoservers: http://143.93.114.104/geoserver

Die Layervorschau[386] (vgl. Abbildung 6-24) zeigt mögliche Darstellungsformen, sowohl als WMS (z.B. PNG zur Einbindung in Leaflet), sowie als GML (in verschiedenen Versionen) und GeoJSON.

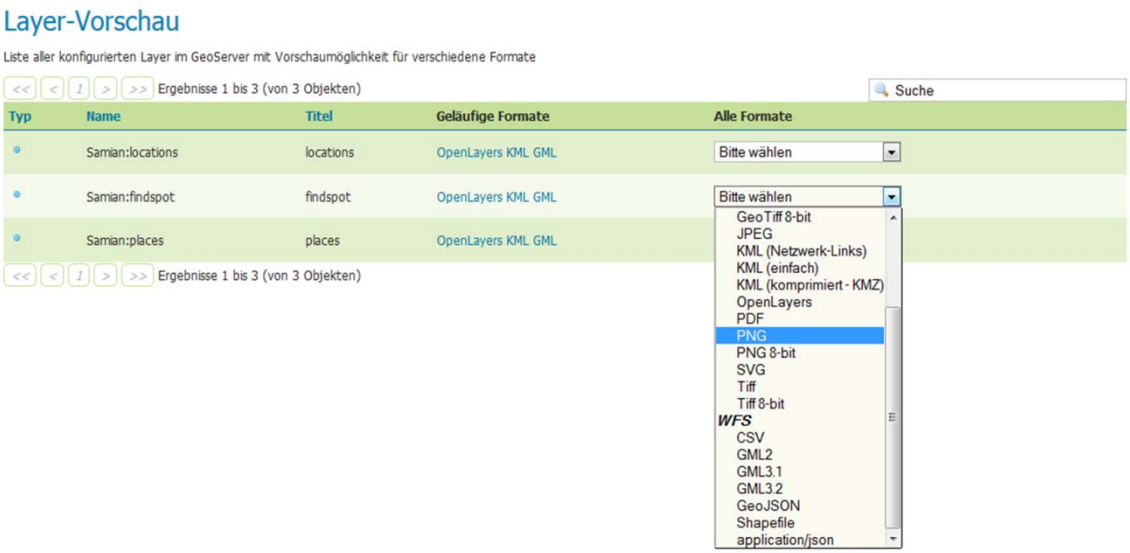


Abbildung 6-24: Geoserver (Layervorschau)

Ein Web Map Service[387] nach OGC besitzt drei Funktionen, welche von einem Benutzer mittels Hyper Text Transfer Protocol (HTTP[388]) angefragt werden können: GetCapabilities (Fähigkeiten des WMS), GetMap (georeferenziertes Bild) und optional getFeatureInfo (Beantwortung von Fragen zu einem Kartenausschnitt), vgl. GISWIKI (2013).

Abbildung 6-25 zeigt einen Ausschnitt eines XML-Response einer getCapabilities-Anfrage inkl. einiger Metadaten wie Kontakt und der zur Verfügung stehenden Layer. Abbildung 6-26 hingegen spiegelt eine getMap-Anfrage des Layers findspots in Open-Layers Repräsentation wider. Hier ist zu erkennen, dass sowohl eine getFeatureInfo (Abbildung unten), wie auch eine farbliche und symbolische Unterscheidung anhand verschiedener Attribute (rot:kilnsite, grün:no kilnsite) möglich ist. Diese Styles können mit Hilfe der Styled Layer Desciption[389] (SLD) beschrieben werden, hier: kilnsites.sld (vgl. Anhang G: SLD der Kilnsites).

---

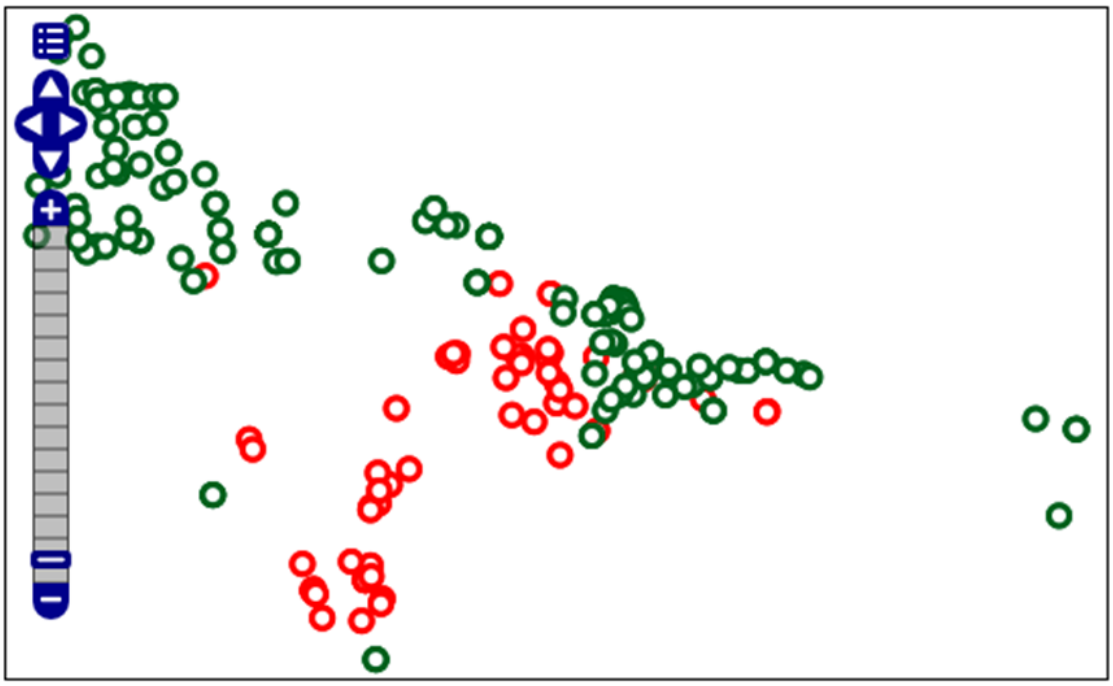
[386] vgl. http://143.93.114.104/geoserver/web/?wicket:bookmarkablePage=:org.geoserver.web.demo.MapPreviewPage

[387] vgl. http://www.opengeospatial.org/standards/wms (abgerufen 04.12.2013)

[388] vgl. http://tools.ietf.org/html/rfc2616 (abgerufen 04.12.2013)

[389] vgl. http://www.opengeospatial.org/standards/sld (abgerufen 04.12.2013)

Abbildung 6-25: GetCapabilities (Beispiel-XML)[390]



Scale = 1 : 18M 4.27869, 50.21282

findspot

fid	name	lat	lon	datemin	datemax	kilnsite
findspot.1	Aardenburg	51.28333282	3.450000048	100	270	false

Abbildung 6-26: GetMap (OpenLayers)[391]

[390] vgl. http://143.93.114.104/geoserver/Samian/wms?service=WMS&request=GetCapabilities

[391] vgl. http://143.93.114.104/geoserver/Samian/wms?service=WMS&version=1.1.0&request=GetMap&layers=Samian:findspot&styles=&bbox=-4.3116660118103,42.3166656494141,19.0833358764648,56.5413513183594&width=542&height=330&srs=EPSG:4326&format=application/openlayers

Analog zum WMS stellt der Web Feature Service (WFS) ebenfalls mehrere Funktionen[392] bereit. Insbesondere sei hier auf getCapabilities[393] (Fähigkeiten des WFS), describeFeatureType (Informationen zur Struktur der einzelnen Feature Types) und getFeature[394] (Rückgabe der einzelnen Features, z.B. als GML oder GeoJSON) verwiesen. Abbildung 6-27 zeigt den describeFeatureType XML-Response einer Anfrage des Layers feature im Arbeitsbereich Samian. Hier ist zu erkennen, dass die Attribute des Findspots als element verfügbar sind und somit an dieser Stelle Semantik in die Daten eingebracht wird.

```
- <xsd:schema elementFormDefault="qualified" targetNamespace="Samian">
    <xsd:import namespace="http://www.opengis.net/gml" schemaLocation="http://143.93.114.104:80/geoserver/schemas/gml/2.1.2/feature.xsd"/>
  - <xsd:complexType name="locationsType">
    - <xsd:complexContent>
      - <xsd:extension base="gml:AbstractFeatureType">
        - <xsd:sequence>
            <xsd:element maxOccurs="1" minOccurs="0" name="id" nillable="true" type="xsd:int"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="name" nillable="true" type="xsd:string"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="lat" nillable="true" type="xsd:double"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="lon" nillable="true" type="xsd:double"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="datemin" nillable="true" type="xsd:int"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="datemax" nillable="true" type="xsd:int"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="kilnsite" nillable="true" type="xsd:boolean"/>
            <xsd:element maxOccurs="1" minOccurs="0" name="geom" nillable="true" type="gml:PointPropertyType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
    <xsd:element name="locations" substitutionGroup="gml:_Feature" type="Samian:locationsType"/>
  </xsd:schema>
```

Abbildung 6-27: DescribeFeatureType[395]

Zur Einbindung von Features in Webkartenanwendungen, wie z.B. Leaflet oder Open Layers[396], ist ein Web Feature Service (z.B. mit der Rückgabe von GeoJSON) nötig. Der zuvor gezeigte und erstelle WFS des Geoservers bietet jedoch keine Rückgabe eines GeoJSON-Objekts nach einer jQuery[397]-Anfrage via Ajax[398] mittels JavaScript an. Des Weiteren bildet die Same Domain Policy[399] eine weitere Hürde, welches das Auslesen des Response verhindert. Aufgrund dessen dient ein Proxy[400], der auf dem gleichen Server wie die Anwendung läuft, als Hilfsprogramm, welches eine

---

[392] vgl. http://www.giswiki.org/wiki/Web_Feature_Service (abgerufen 04.12.2013)

[393] siehe z.B. http://143.93.114.104/geoserver/Samian/wfs?service=WFS&request=GetCapabilities

[394] siehe z.B. http://143.93.114.104/geoserver/Samian/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=Samian:locations&maxFeatures=50&outputFormat=json

[395] siehe z.B. http://143.93.114.104/geoserver/Samian/ows?service=WFS&version=1.0.0&request=describefeaturetype&typeName=Samian:locations&maxFeatures=50&outputformat=xmlschema

[396] vgl. http://openlayers.org (abgerufen 04.12.2013)

[397] vgl. http://jquery.com (abgerufen 04.12.2013)

[398] vgl. http://de.wikipedia.org/wiki/Ajax_%28Programmierung%29 (abgerufen 04.12.2013)

[399] vgl. http://jsonp.eu/sop.html (abgerufen 04.12.2013)

[400] vgl. http://de.wikipedia.org/wiki/Proxy_%28Rechnernetz%29 (abgerufen 04.12.2013)

Anfrage an den Geoserver stellt und ein GeoJSON-Objekt als Antwort bietet. Der Proxy wird hier als getfeature.war im Apache Tomcat abgelegt und kann via http://143.93.114.104/getfeature angesprochen werden. Dem Servlet findspots kann zudem ein Parameter "param" mit folgenden Werten übergeben werden[401]: places, kilnsite, locations, export und all. Der Parameter steuert die Rückgabe und filtert die Anfrage an den Geoserver, so dass beispielsweise nur Places aus der Datenbank gelesen und als GeoJSON-Objekt zur Verfügung gestellt werden.

Der Proxy 'getfeature' besteht aus einer JAVA Klasse (GeoserverConnection), sowie einem Servlet (getFeature). Das Servlet dient hier zur Steuerung des Requests und des Response und verwaltet als zentrale Stelle die Beschaffung des JSON Objekts vom GeoServer, die Klasse stellt die Verbindung zum Geoserver (evtl. mit Filterung) her. Der grobe Ablauf ist in Abbildung 6-28 aufgezeigt.

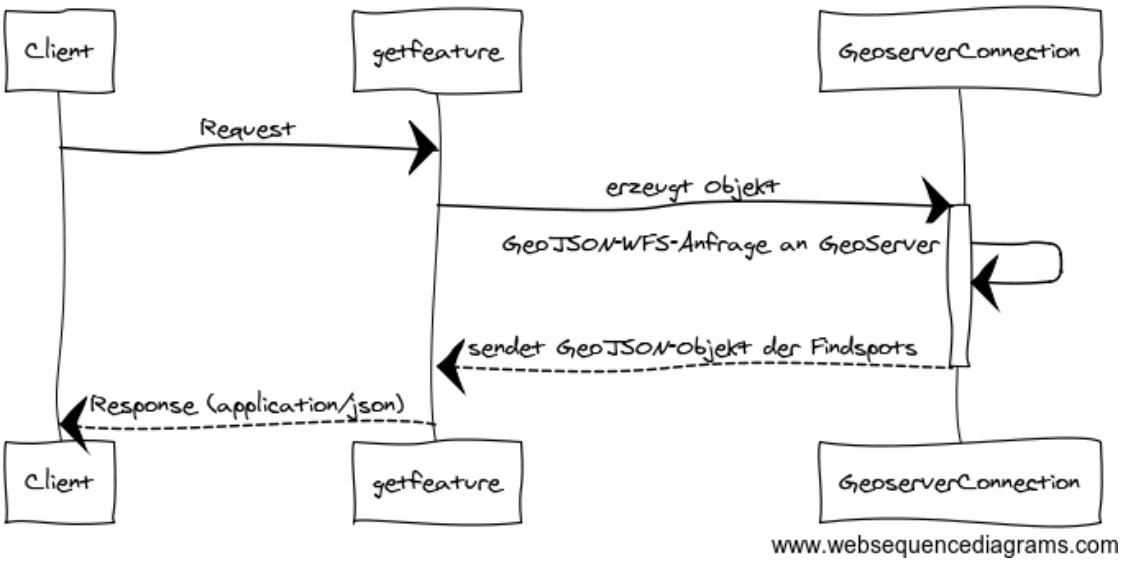


Abbildung 6-28: Proxy getFeature (Sequenzdiagramm)

[401] z.B. http://143.93.114.104/getfeature/findspots?param=places

Nach einem Request des Client an das Servlet getfeature wird zunächst der Parameter ausgelesen und mit Hilfe dessen eine Instanz der Klasse GeoServerConnection erstellt. Diese öffnet eine HTTP URL Connection zum Geoserver und erstellt einen WFS Request[402]. Dieser Request kann zur Anfrage einer bestimmten Gruppe (z.B. nur Kilnsites) einen Filter[403] enthalten[404]. Der WFS Response wird als String empfangen und vom Servlet als application/json Objekt zu Client zurückgesendet, so dass dieses in Webanwendungen wie z.B. Leaflet eingebunden werden kann.

6.3 Triple Store

Kapitel 6.1.5 zeigt die Erstellung und Repräsentationsformen der Findspots, Fragments und Potter als Linked Data in RDF-Syntax. In Kapitel 6.1.8 werden Dateien beschrieben, die einen semantischen Dump[405] der Datenbank darstellen. Dieser dient als Grundlage zur Befüllung des Sesame[406] Triplestores[407] (vgl. Kapitel 2.5.5), welcher ebenfalls als Anwendung[408] im Apache-Tomcat-Container zur Verfügung steht. Der Sesame Server[409] besitzt zur vereinfachten Anwendung ein Benutzerinterface[410], vgl. Abbildung 6-29. Ausführliche Informationen zu Triplestores sind [HAFT, 2013] zu entnehmen. Zur Speicherung der einzelnen Samian-Triples dient im Rahmen dieser Arbeit das Repository 'samian[411]'. Die Daten werden im Arbeitsspeicher vorgehalten und in keiner weiteren Datenbank abgelegt.

---

[402] z.B. bei param=all
http://143.93.114.104/geoserver/Samian/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=Samian:findspot&outputFormat=json

[403] vgl. http://docs.geoserver.org/latest/en/user/filter/function_reference.html (abgerufen 04.12.2013)

[404] z.B. bei param=kilnsites
http://143.93.114.104/geoserver/Samian/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=Samian:findspot&outputFormat=json&Filter=%3CFilter%3E%3CPropertyIsEqualTo%3E%3CPropertyName%3Ekilnsite%3C/PropertyName%3E%3CLiteral%3Etrue%3C/Literal%3E%3C/PropertyIsEqualTo%3E%3C/Filter%3E

[405] http://de.wikipedia.org/wiki/Dump (abgerufen 04.12.2013)

[406] vgl. http://www.openrdf.org (abgerufen 04.12.2013)

[407] vgl. http://www.w3.org/2001/sw/wiki/Category:Triple_Store (abgerufen 04.12.2013)

[408] Anwendungsszenarien:
http://openrdf.callimachus.net/sesame/2.7/docs/users.docbook?view#Getting_Started

[409] Sesame Server: http://143.93.114.104/openrdf-sesame (abgerufen 04.12.2013)

[410] Sesame Workbench: http://143.93.114.104/openrdf-workbench (abgerufen 04.12.2013)

[411] vgl. http://143.93.114.104/openrdf-sesame/repositories/samian

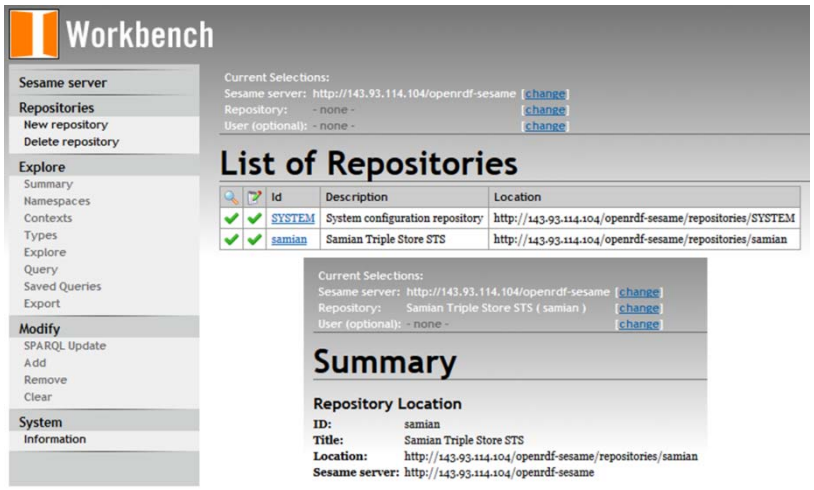


Abbildung 6-29: Open Sesame Benutzerinterface

Alle erstellten Triples der ReST-Schnittstelle liegen in drei Dateien vor:

- http://143.93.114.104/share/potters.ttl
- http://143.93.114.104/share/fragments.ttl
- http://143.93.114.104/share/findspots.ttl

Mit Hilfe des Workbench (Modify →Add →Datei auswählen) können die Files in den Triplestore übertragen werden (vgl. Abbildung 6-30). Jede Datei erhält seinen eigenen Context, sodass z.B. alle Triples eines Contexts gelöscht werden können (vgl. Abbildung 6-30, unten). Die Samian-Datenbank enthält 1.044.302 Tripel (Stand: 19.11.2013).

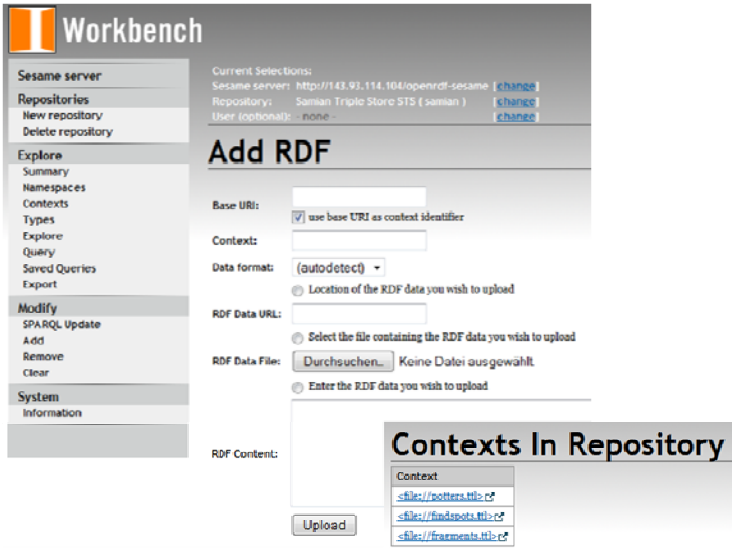


Abbildung 6-30: Triple in Sesame einfügen

Die Funktion Explore →Query ermöglicht die Abfrage des Triplestores mit Hilfe der SPARQL-Abfragesprache (vgl. Kapitel 2.5.4). Die Anzeige aller im Triplestore enthaltenen Triple ist mit nachfolgender Abfrage möglich (vgl. auch Abbildung 6-31).

```
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sys:<http://www.openrdf.org/config/repository#>
SELECT * WHERE { ?s ?p ?o }
```

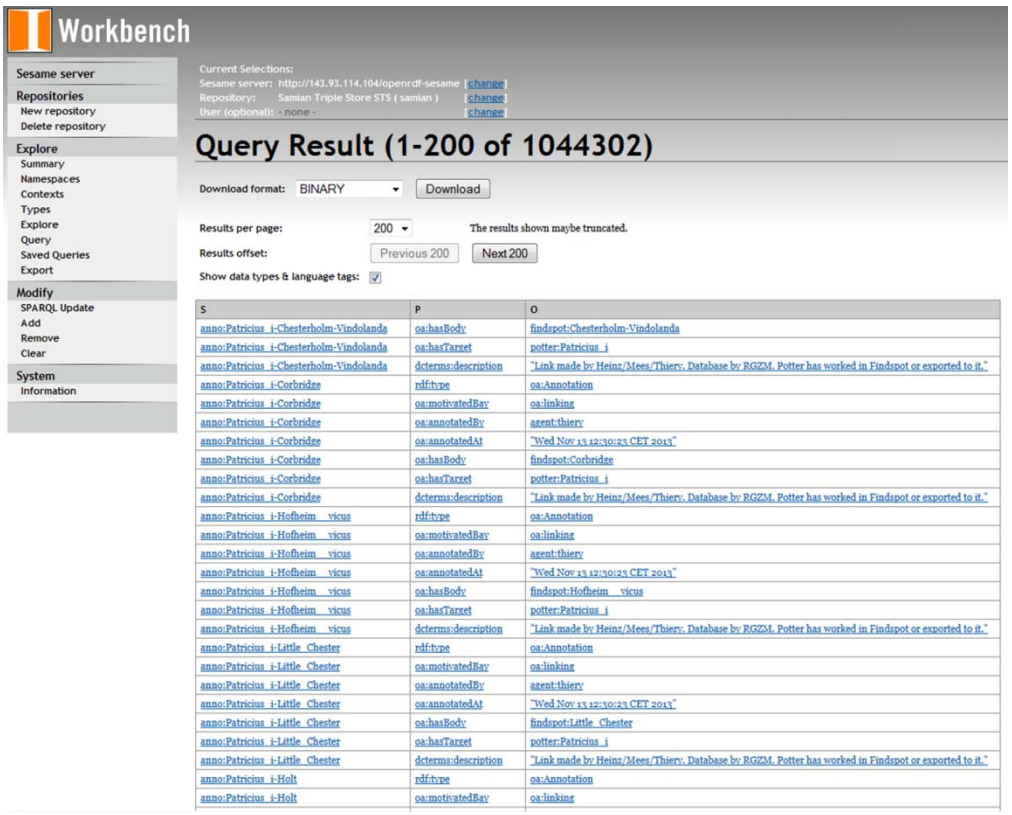


Abbildung 6-31: GeInArFa Triple Liste

Als SPARQL-Entry-Point und damit die zentrale Stelle zur Nutzung des Triplestores von Dritten (auch von internen Programmen auf dem GeInArFA Server), dient nachfolgende URL. Bei einer Anfrage aus JAVA wird ein SPARQL Query Results XML[412] erzeugt, welches mit üblichen XML Parsern durchsucht werden kann.

```
http://143.93.114.104/openrdf-
workbench/repositories/samian/query
?action=exec&queryLn=SPARQL&query={query}&limit=100&infer=true
```

[412] vgl. http://www.w3.org/2005/sparql-results# und
http://www.w3.org/TR/rdf-sparql-XMLres (abgerufen 04.12.2013)

In der HTTP-GET-Anfrage sind das verwendete Repository (samian), die Anfragesprache (SPQRQL), die query und die Anzahl der Ergebnisse (limit=100) zu beachten. Anstelle der {query} folgt eine SPARQL Abfrage inkl. Prefixes, wobei hier auf eine Maskierung[413] der Sonderzeichen in der URL zu achten ist.

6.4 Beurteilung

Die Datenbereitstellung kann in drei große Bereiche eingeteilt werden: die ReST-Schnittstelle, OGC-Services und den Sesame Triplestore. Die verwendete PostGIS Datenbank ermöglicht die unproblematische Bereitstellung der (Geometrie-)Daten über den Geoserver in standardisierten Formaten wie WFS und WMS. Die Nutzung des Sesame Triplestores ist ebenfalls als nicht kritisch anzusehen. Eine Befragung dessen, mit Hilfe von SPARQL, ist über eine ReST-Schnittstelle möglich, über welche ebenfalls Daten gesendet werden könnten. Für die Zwecke dieser Arbeit (Durchsuchen des RDF Graphen und Verknüpfung mit Pelagios Datasets) ist der vorliegende Triplestore somit völlig ausreichend. Dahingegen stellt die Bereitstellung der Daten für den Triplestore, das heißt die Erzeugung der Triple und der semantischen Abbildungen (z.B. XML) eine gewisse Problematik dar. Wie gezeigt, ist es für die drei hier benutzten Objektarten (Potter, Findspot, Fragment) schwierig geeignete (einheitliche) bereits vorhandene XML-, JSON-, bzw. RDF-Repräsentationen anzuwenden. Der Geoserver kann dazu verwendet werden GML- und GeoJSON-Objekte zu erstellen. MIDAS XML würde den gesamten Prozess der Samian Ware abbilden können, eine Modellierung ist daher anzustreben (jedoch Zeit- und Arbeitsintensiv). Pleiades löst die Problematik der Findspots in RDF hinreichend. Die Abbildung von historischen Personen (Töpfer) und Keramikfragmenten als Linked Data kann jedoch als nicht befriedigend bewertet werden. In diesen Fällen sollte darüber nachgedacht werden, eine eigene Ontologie der Samian Ware, inkl. aller in Betracht kommenden Schritte (Erzeugung, Stempelung, Verhandlung, Fund, etc.) zu erzeugen. Gegebenenfalls sollte ein Mapping des gesamten Sachverhalts zu CIDOC CRM erfolgen. Dies stellt jedoch einen erheblichen Aufwand dar, sollte jedoch aufgrund der großen Anzahl an Anwendern in den Geisteswissenschaften favorisiert werden. Verknüpfungen der Objektarten untereinander, sowie die Verbindung zu Pleiades Places werden mit Open Annotation gelöst. Die Entwicklung eines eigenen (eindeutigen) Vokabulars dieser Beziehungen würde die Problematik der nicht expliziten Beziehung zweier mit Open Annotation verbundenen Ressourcen eindämmen. Hierbei liegt die Problematik jedoch in den oftmals nicht eindeutig beschreibbaren Beziehungen. Dieser Umstand müsste ebenfalls berücksichtigt werden.

[413] vgl. http://schneegans.de/asp.net/url-escape (abgerufen 04.12.2013)

7 Datenmapping und Datenverknüpfung

Durch die in Kapitel 6 beschriebene ReST-Schnittstelle werden aktuelle Technologien interoperabler Datenhaltung (z.B. XML, JSON), sowie der semantischen Modellierung (RDF) archäologischer Informationen aufgezeigt und URIs zu Fragmenten römischer Keramik, deren Fund- und Herstellungsorte, sowie deren Hersteller zur Verfügung gestellt. Um das Potenzial der Verlinkungen heterogener Informationen über Linked Data herauszuarbeiten, dient die Pelagios-Schnittstelle. Die räumliche Verknüpfung anderer Objektgattungen (z.B. Münzen) erfolgt hierbei über Pleiades Places. Hierzu sind die Fundorte der GeInArFa ReST-Schnittstelle auf die entsprechenden Places des Pleiades-Projekts zu mappen. Die folgenden Kapitel erläutern das Konzept der Pleiades Places, die PELAGIOS API, das vorgenommene manuelle Mapping, sowie die Pelagios Connection API, die als Resultat des Prozesses angesehen werden kann.

7.1 Pleiades und Pleiades Places

Pleiades ist ein historischer Gazeteer und gibt Wissenschaftlern die Möglichkeit historische geografische Informationen über die antike Welt zu nutzen, zu erstellen und zu teilen. Er verbindet Namen und Positionen im Kontext ihrer jeweiligen Zeit. Pleiades stellt die Informationen zu Orten als Linked Data zur Verfügung, vgl. Kapitel 3 [PLEIADES, 2013A] und ist ein Gemeinschaftsprojekt des Ancient World Mapping Center[414], dem Stoa Concortium[415] und des Institute for the Study of the Ancient World[416]. Zum Stichtag des 27. Oktober 2013 enthielt Pleiades 34.731 historische Places, 30.156 historische Namen und 38.532 historische Locations [PLEIADES, 2013C].

Pleiades unterscheidet zwischen Places, Names und Locations. Ein Place ist ein geographischer und historischer Kontext für Namen und Locations. Places können Merkmale der physischen Welt enthalten (z.B. ein Meer, eine Bucht, eine Straße, etc.), sie sind jedoch auf menschliche Erfahrungswerte aufgebaut[417] und können sich Lauf der Zeit (auch geografisch) verändern. Darüber hinaus kann ein Place auch ohne Namen, geografische Position oder als fiktiver mystischer Ort existieren. Locations sind aktuelle oder ehemalige konkrete räumliche Einheiten (z.B. die Mittellinie eines Flussbettes) und gehören zu genau einem Ort. Ein Name ist eine aktuelle oder ehemalige abstrakte textuell Entität. Ein Name gehört ebenfalls nur zu einem Place [GILLIES, 2011]. Der schematische Aufbau dieses Dreigestirns als UML-Diagramm ist Abbildung 7-1 zu entnehmen.

---

[414] vgl. http://www.unc.edu/awmc (abgerufen 04.12.2013)

[415] vgl. http://www.stoa.org (abgerufen 04.12.2013)

[416] vgl. http://isaw.nyu.edu (abgerufen 04.12.2013)

[417] Y. Tuan, "Place: An Experiential Perspective," Geographical Review, vol. 65, Apr. 1975, pp. 151-165.

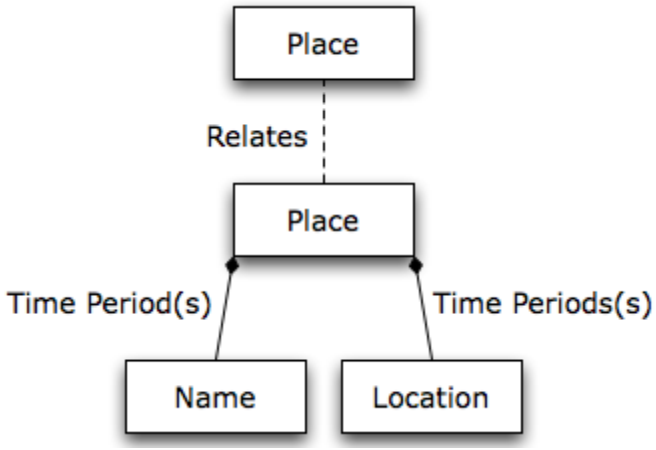


Abbildung 7-1: Pleiades Places (UML) [PLEIADES, 2013B]

Das UML Diagramm zeigt den Zusammenhang zwischen den einzelnen Entitäten auf: Namen und Locations sind Tochterklassen der Vaterklasse Place mit einer Komposition[418]. Zeiten sind als Attribute eines Namen oder einer Location modelliert. Places können sich im Vergleich zu Namen und Locations auf andere Place-Objekte beziehen [GILLIES, 2011].

Zur Gegenüberstellung des in Pleiades vorgestellten Konzepts mit dem Samian-Datenkonzept (vgl. Kapitel 5.1) dient das Beispiel Langenhain, vgl. Abbildung 7-2. Der Pleiades Place Langhain[419] besitzt ID 109100, sowie eine Location[420] jedoch keinen Namen, ist den Typen fort und tower zugeordnet und besitzt keine geografische Position. Seiner Location 'DARMC location 6131' ist eine Koordinate, seine zeitliche Einordnung, jedoch kein Type zugeordnet. Im Vergleich dazu besitzt der Samian Place die ID Langenhain[421], eine Koordinate, einen Namen, eine zeitliche Einordnung, eine Location, sowie andere Types. Die Location Langenhain__Store zeichnet sich ebenfalls durch eine geographische und zeitliche Einordnung, einen Namen und verschiedene Typen aus. Vergleicht man beide Konzepte, werden Unterschiede in der Zuordnung von Namen, Zeiten und Koordinaten deutlich. Der kleinste gemeinsame Nenner ist die kompositive Beziehung der Places und Locations. Dies ist somit der Ansatzpunkt zur Konzeptentwicklung eines Datenmapping zwischen Pleiades und Samian Places. Das Konzept von Pleiades schlägt vor, dass Places Beziehungen zu anderen Orten haben können, Locations jedoch nicht. Auf Grund dieses Vorschlags wird in den folgenden Kapiteln ein Datenmapping nur zwischen Pleiades und Samian Places vorgenommen.

---

[418] vgl. http://openbook.galileocomputing.de/oop/oop_kapitel_04_003.htm#mj8118cb8c892c6648834fb351f4be17aa (abgerufen 04.12.2013)

[419] vgl. http://pleiades.stoa.org/places/109100 (abgerufen 04.12.2013)

[420] vgl. http://pleiades.stoa.org/places/109100/darmc-location-6131 (abgerufen 04.12.2013)

[421] vgl. http://143.93.114.104/rest/samian/findspots/Langenhain

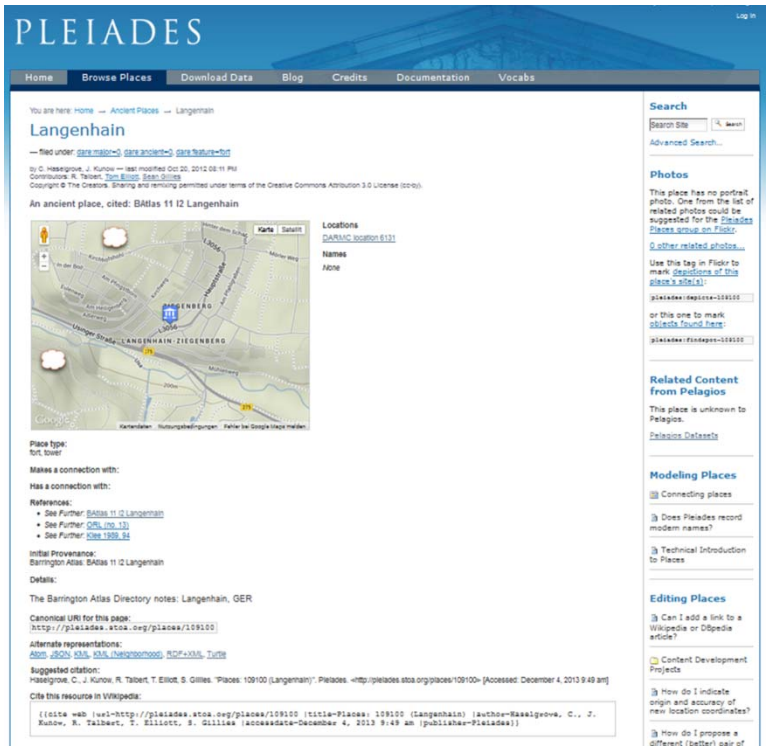


Abbildung 7-2: Pleiades Place Langenhain[422]

Alle Pleiades Ressourcen können als CSV, KML oder RDF aus verschiedenen Dumps zu eigenen Weiterverarbeitung aus dem Internet heruntergeladen werden [GILLIES, 2013B]. Des Weiteren ist ein MYSQL-Dump[423] aller Pleiades Places (Stichtag: 30. Juni 2013) verfügbar, welcher in die PostGIS-Datenbank integriert ist (Tabelle placrefer) und zum Datenmapping (vgl. Kapitel 7.3) genutzt wird.

7.2 Pelagios und die Pelagios API

Pelagios[424] (Pelagios: Enable Linked Ancient Geodata In Open Systems) ist ein Projektkollektiv[425] aus Projekten verschiedenster historischer Bereiche. Pelagios' Ziel ist das Aufzeigen von Potentialen der Linked Data am Beispiel von antiken Orten verschiedener Ressourcen zur Generierung neuer Methoden der Erforschung und Visualisierung[426] für Wissenschaftler und der breiten Öffentlichkeit, vgl. auch Kapitel 3 [PELAGIOS, 2013]. Des Weiteren soll Wissenschaftlern die Möglichkeit gegeben werden, Referenzen zu antiken Orten in Karten, Texten, Bildern und Tabellen unter Einbeziehung der Pleiades Places zu erstellen [BARKER, 2011]. Pelagios ist daher außerordentlich gut

---

[422] Quelle: http://pleiades.stoa.org/places/109100

[423] vgl. http://pelagios-project.blogspot.co.uk/2013/07/a-sql-version-of-pleiades-dataset.html (04.12.2013)

[424] vgl. http://pelagios-project.blogspot.de (abgerufen 04.12.2013)

[425] Projektpartner: http://pelagios-project.blogspot.de/p/about-pelagios.html (abgerufen 04.12.2013)

[426] aktuelle Anwendungen: http://pelagios-project.blogspot.de/p/pelagios-in-use.htm (abgerufen 04.12.2013)

geeignet durch eine Verlinkung der Samian Places zu Pleiades Places von der großen Datenvielfalt der Pelagios Partner zu profitieren und neue Erkenntnisse zu gewinnen.

Kapitel 7.1 zeigt das Beispiel des Pleiades Place Langenhain[427]. Langenhain ist ebenfalls mit dem Pelagios-Projekt verbunden[428], vgl. Abbildung 7-3 und zeigt Verbindungen mit zwei Datasets von Pelagios-Partnern auf.

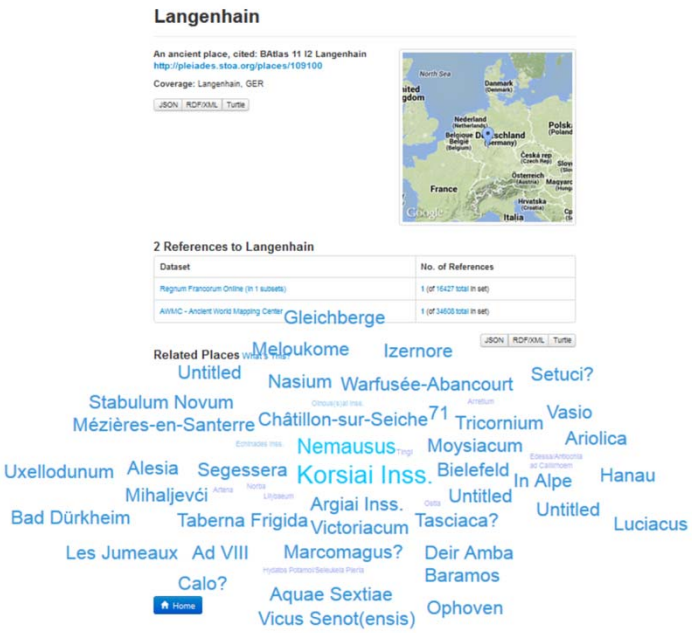


Abbildung 7-3: Pelagios Place Langenhain

Mit Hilfe eines Mapping zwischen einem Samian und Pleiades Place (vgl. Kapitel 7.3) ist es somit möglich, Datasets des Pelagios-Projekts auf den eigenen Samian Place zu referenzieren. Hierzu sind zwei Varianten denkbar: 1. Nutzung des Triplestores mit der Speicherung eines Dumps der Samian und aller Pelagios Verlinkungen zu einem Stichtag oder 2. Nutzung einer API, welche durch Eingabe einer Pleiades ID immer die aktuellsten Datasets des Pelagios Projekts zurückgibt. Auf Grund der Aktualität der Daten (werden neue Projekte an Pelagios gelinkt, werden neue Informationen über den Place automatisch hinzugefügt) und der nicht benötigten Datenhaltung aller Pelagios Links wird Variante zwei genutzt, da Pelagios eine API[429] für diese Belange zur Verfügung stellt.

Die Pelagios HTTP API ermöglicht eine einfache Suche durch das Pelagios Netzwerk von Ortsreferenzen. Sie verfügt über eine einfache HTML-Schnittstelle und bietet JSON/JSONP bzw. RDF (XML oder Turtle) Responses und stellt nur lesende Funktionen (HTTP GET) zur Verfügung

---

[427] vgl. http://pleiades.stoa.org/places/109100

[428] http://pelagios.dme.ait.ac.at/api/places/http%3A%2F%2Fpleiades.stoa.org%2Fplaces%2F109100 (abgerufen 04.12.2013)

[429] vlg. https://github.com/pelagios/pelagios-cookbook/wiki/Using-the-Pelagios-API (abgerufen 04.12.2013)

[GITHUB, 2012]. Eine Test-Instanz[430] der API ist ebenfalls verfügbar und wird im Rahmen dieser Masterarbeit genutzt. Im Folgenden werden die Anfrage-Möglichkeiten und Antworten der API (anhand des Beispiels Langenhain) vorgestellt, welche in der Pelagios Connection API (vgl. Kapitel 7.4) genutzt werden.

Die Pelagios API nutzt drei generelle Typen: Places, Datasets[431] und Annotationen. Jedem Place können Datasets eines Pelagios-Partners durch die Pleiades PlaceID zugeordnet werden, darüber hinaus enthält jedes Dataset eine Annotation, in der der Link zur Ressource abrufbar ist. Durch die API können zum einen Informationen zu allen Places (/places), zu einem bestimmten Place mit einer Pleiades ID (/places/{id}) und die die Datasets eines bestimmten Places erhalten werden (/places/{id}/datasets). Zum anderen können alle Datasets (/datasets), Informationen zu einem bestimmten Dataset (/datasets/{id}) und Annotationen zu einem bestimmten Dataset (/datasets/{id}/annotations) abgefragt werden [GITHUB, 2012].

Eine Place-Referenz wird immer durch Angabe der Pleiades URL angegeben (z.B. http://pleiades.stoa.org/places/109100). Die Datasets des Places Langenhain können somit durch folgende URL (/places/{PleiadesURI}/datasets) in Turtle Syntax abgerufen werden[432]:

```
http://pelagios.dme.ait.ac.at/api/places/
http%3A%2F%2Fpleiades.stoa.org%2Fplaces%2F109100/datasets.ttl
```

Die Antwort[433] enthält zwei Datasets (Regnum Francorum Online und AWMC - Ancient World Mapping Center):

- http://pelagios.dme.ait.ac.at/api/datasets/300b9dc90cfc0d61ad9e0a7ebedf6207
- http://pelagios.dme.ait.ac.at/api/datasets/b7a03862acd3033c08bd06926e60b742

Die Ressourcen, welche hinter den Datasets verborgen sind, können durch die Anfrage nach Annotationen (/datasets/{datasetID}/annotations) erhalten werden[434]:

---

[430] vgl. http://pelagios.dme.ait.ac.at/api (abgerufen 04.12.2013)

[431] vgl. http://pelagios.dme.ait.ac.at/api/datasets (abgerufen 04.12.2013)

[432] vgl. auch https://github.com/pelagios/pelagios-cookbook/wiki/API-Details%3A-Show-Datasets-for-Place (abgerufen 04.12.2013)

[433] vgl. http://pelagios.dme.ait.ac.at/api/places/http%3A%2F%2Fpleiades.stoa.org%2Fplaces%2F109100/datasets.ttl (abgerufen 04.12.2013)

[434] vgl. auch https://github.com/pelagios/pelagios-cookbook/wiki/API-Details%3A-Show-Annotations-for-Dataset (abgerufen 04.12.2013)

http://pelagios.dme.ait.ac.at/api/
datasets/b7a03862acd3033c08bd06926e60b742/
annotations.json?
forPlace=http%3A%2F%2Fpleiades.stoa.org%2Fplaces%2F109100

http://pelagios.dme.ait.ac.at/api/
datasets/300b9dc90cfc0d61ad9e0a7ebedf6207/
annotations.json?
forPlace=http%3A%2F%2Fpleiades.stoa.org%2Fplaces%2F109100

Abbildung 7-4 zeigt den JSON-Response der Annotation des Datasets b7a03862acd3033c08bd06926e60b742 für den Pleiades Place 109100 (Langenhain) und damit den Link via Open-Annotation-Target (vgl. Kapitel 3): http://awmc.unc.edu/api/omnia/162088.


```
{
    "dataset": "http://pelagios.dme.ait.ac.at/api/datasets
/b7a03862acd3033c08bd06926e60b742",
    "place": "http://pleiades.stoa.org/places/109100",
    "total_count": 1,
    "display_offset": 0,
    "display_limit": 1,
    "annotations":
    [
        {
            "uri": "http://awmc.unc.edu/api/rdf_main.rdf#awmc_162088",
            "title": "Langenhain",
            "hasBody": "http://pleiades.stoa.org/places/109100",
            "hasTarget": "http://awmc.unc.edu/api/omnia/162088",
            "in_dataset":
            {
                "uri": "http://pelagios.dme.ait.ac.at/api/datasets
/b7a03862acd3033c08bd06926e60b742",
                "title": "AWMC - Ancient World Mapping Center",
                "annotations_total_in_set": 34608
            }
        }
    ]
}
```


Abbildung 7-4: JSON Response der Pelagios API

7.3 Pleiades Mapping

Wie bereits in Kapitel 7.2 vorgestellt, bietet Pelagios eine API, welche Datasets und Annotationen zu bereits verlinkten Ressourcen mit Hilfe der Pleiades ID im Netzwerk von Pelagios sucht. Mappt man nun die eigenen Samian Places mit Pleiades Places, können mit Hilfe einer eigenen API (vgl. Kapitel 7.4), zu einem Fundort, Datasets und Annotationen des Pelagios-Projekts gefunden und somit die eigenen Daten um weitere bereichert werden. Dieses Mapping ist nicht automatisch möglich. Geisteswissenschaftliche Daten zeichnen sich durch eine Fülle von inhärenten Informationen aus, welche nicht durch rein logisch operierende Programme erfasst werden können. Ein gleicher Name oder eine Ähnlichkeit in der Position ist noch kein Indiz für die Gleichheit zweier Ressour-

cen. Aufgrund dessen wird dem Geisteswissenschaftler ein Tool zur Verfügung gestellt, das ihm die Arbeit des Mapping erleichtert.

In dieser Masterarbeit erfolgt eine räumliche Abfrage der Distanz zwischen jedem Samian Place und allen Pleiades Places (nur Places werden mit Places verbunden, vgl. Kapitel 7.1). Die jeweils fünf am nächsten liegenden Pleiades Places werden aufgelistet und können mit der Distanz (in Metern) und einem visuellen Namensvergleich dem Mapping eine Hilfestellung leisten. Zudem wird ein Link zum Pleiades Place angegeben. Diese genauen Beschreibungen der Pleiades-Ressource können das Mapping vereinfachen oder verwerfen.

Die Findspots liegen bereits in der PostGIS Datenbank vor. Alle Pleiades Places sind in einem MYSQL-Dump[435] (Stichtag: 30.06.2013) verfügbar[436]. Nach einer manuellen Transformation des Dumps in ein PostGIS lesbares Format stehen die Daten in der Tabelle placrefer zu Verfügung, vgl. Anhang H: Liste der Mapping Ergebnisse.

Das Pleiades Mapping Tool ist als pleiadesMappingTool.war im Apache Tomcat abgelegt und kann mit `http://143.93.114.104/pleiadesMappingTool` gestartet werden. Das Tool beinhaltet ein Servlet: Pleiades, dem kein Parameter übergeben werden kann. Mit Hilfe des Maven Dependency postgresql (Version: 9.1-901.jdbc3[437]) wird der Treiber zur Ansprache der PostGIS Datenbank eingebunden.

Das Tool besteht aus einer Java-Klasse (PostGIS) und einem Servlet. Das Servlet verwaltet den Request und Response (text/plain) und ermittelt mit Hilfe der Klasse PostGIS die gesuchten fünf am nächsten liegenden Pleiades Places. Der Ablauf ist Abbildung 7-5 zu entnehmen.

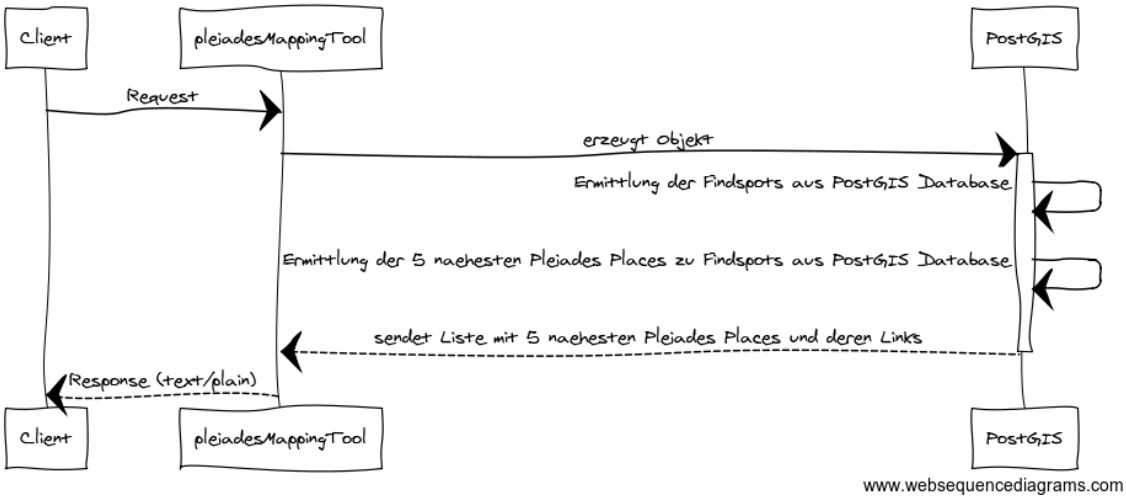


Abbildung 7-5: PleiadesMappingTool (Sequenzdiagramm)

[435] http://pelagios-project.blogspot.co.uk/2013/07/a-sql-version-of-pleiades-dataset.html (04.12.2013)

[436] https://docs.google.com/file/d/0B9YdgOKdbi4VTm1pN3NiRlU5bGM/edit?usp=sharing (04.12.2013)

[437] http://mvnrepository.com/artifact/postgresql/postgresql/9.1-901.jdbc3 (abgerufen 04.12.2013)

Nach dem Request des Client wird eine Instanz der PostGIS Klasse erzeugt. Via SQL Anfrage an die Datenbank (`SELECT findspot.name FROM findspot`) werden alle Findspots in den Zwischenspeicher übertragen. Zur Ermittlung der fünf nächsten Places der Findspots in der Tabelle placrefer werden weitere SQL Anfragen an die Datenbank gesendet. Die Geometrien der Samian und Pleiades Places liegen in WGS84 (EPSG:4326[438]) vor. Eine räumliche Abfrage in diesem Referenzsystem würde auf Grund seiner Definition keine metrische Größe ergeben. Ein interpretierbarer metrischer Wert ist beispielsweise in kartesischen Koordinatensystemen wie Gauß-Krüger[439] oder UTM[440] zu erreichen. Infolgedessen wird das System WGS84/UTM32 (EPSG:32632[441]) genutzt. UTM32 liegt in der Mitte Deutschlands und somit im Zentrum der Findspots. Aufgrund der zentralen Lage ist nur eine minimale (für diese Anwendung nicht bemerkenswerte) Verzerrung in den Distanzen, außerhalb des definierten Bereichs der Breitengrade 6° und 12°, vorhanden. Die genutzte SQL Anfrage transformiert die in der Datenbank gespeicherten WGS84 Geometrien in UTM32 und berechnet die fünf nächsten Distanzen.

```
SELECT
ST_Distance(ST_Transform(p.geom, 32632),
ST_Transform(f.geom , 32632))
AS dist, p.place_name, p.era, p.id
FROM
placrefer p, findspot f WHERE f.name = '{findspot}'
ORDER BY
ST_Distance(ST_Transform(p.geom, 32632),
ST_Transform(f.geom , 32632))
ASC LIMIT 5
```

Nach Beendigung der Abfrageserie wird das Ergebnis (Name des Place, Distanz zum findspot, Ära des Places, Pleiades ID und Link, sowie ein Match, wenn die Distanz < als 5km ist) an den Client zurückgegeben, vgl. Abbildung 7-6.

---

[438] vgl. http://spatialreference.org/ref/epsg/4326/ (abgerufen 04.12.2013)

[439] vgl. http://henrik-seidel.gmxhome.de/gausskrueger.pdf (abgerufen 04.12.2013)

[440] vgl. http://vermessung.bayern.de/file/pdf/1910/UTMAbbildungundKoordinaten.pdf (04.12.2013)

[441] vgl. http://spatialreference.org/ref/epsg/32632/ (abgerufen 04.12.2013)

```
Findspot(Place): Aardenburg
Place                                    Distance                 Era    ID          Pleiades-Link                                    Match(<5km)
-----------------------------------------------------------------------------------------------------------------------------------------
Aardenburg                               1178.3006086829068 4m    RM     108722      http://pleiades.stoa.org/places/108722           true
Brugge                                   17764.442163559455 4m    RL     108830      http://pleiades.stoa.org/places/108830           false
Marsaci?                                 24389.8370992942619m     R      109143      http://pleiades.stoa.org/places/109143           false
Litus Saxonicum                          24389.8370992942619m     L      109117      http://pleiades.stoa.org/places/109117           false
Wenduine                                 25744.1429636312678m     R      109457      http://pleiades.stoa.org/places/109457           false
=========================================================================================================================================

Findspot(Place): Aislingen
Place                                    Distance                 Era    ID          Pleiades-Link                                    Match(<5km)
-----------------------------------------------------------------------------------------------------------------------------------------
Aislingen                                1209.5614660483518m      R      118549      http://pleiades.stoa.org/places/118549           true
Gundremmingen                            3013.1858958987218 1m    RL     118714      http://pleiades.stoa.org/places/118714           false
Ad Novas                                 3694.1455107339606 9m    RL     187253      http://pleiades.stoa.org/places/187253           false
*Raetovarii                              3694.1455107339606 9m    L      118923      http://pleiades.stoa.org/places/118923           false
Sontheim                                 3694.1455107339606 9m    R      118967      http://pleiades.stoa.org/places/118967           false
=========================================================================================================================================

Findspot(Place): Altenstadt
Place                                    Distance                 Era    ID          Pleiades-Link                                    Match(<5km)
-----------------------------------------------------------------------------------------------------------------------------------------
Altenstadt                               28.6841018477337535m     R      118559      http://pleiades.stoa.org/places/118559           true
Glauberg                                 5617.4545839344054 9m    HRL    118701      http://pleiades.stoa.org/places/118701           false
Oberflorstadt                            6391.6045840291590 1m    R      118887      http://pleiades.stoa.org/places/118887           false
Nidderau                                 7369.1502059548211m      R      118873      http://pleiades.stoa.org/places/118873           false
Marköbel                                 7658.3793102162117 1m    R      118834      http://pleiades.stoa.org/places/118834           false
=========================================================================================================================================

Findspot(Place): Aulnay
Place                                    Distance                 Era    ID          Pleiades-Link                                    Match(<5km)
-----------------------------------------------------------------------------------------------------------------------------------------
Au(n)edonnacum                           520.548537279241145m     RL     138200      http://pleiades.stoa.org/places/138200           true
Untitled                                 11836.8670744733772m     RL     141191      http://pleiades.stoa.org/places/141191           false
Untitled                                 11836.8670744733772m     RL     141192      http://pleiades.stoa.org/places/141192           false
Untitled                                 11836.8670744733772m     RL     141215      http://pleiades.stoa.org/places/141215           false
Untitled                                 11836.8670744733772m     RL     141188      http://pleiades.stoa.org/places/141188           false
=========================================================================================================================================
```

Abbildung 7-6: PleiadesMappingTool (Beispielausgabe)

Die in Abbildung 7-6 gezeigte Liste dient als Hilfsmittel (Distanz, Name und Ära) während des Vorgangs des manuellen Mappings. Das Mapping wurde in dieser Arbeit von Allard Mees[442] durchgeführt. Erwartungsgemäß entstehen Probleme bei einem Mapping zweier verschiedener Datensätze. Allard Mees beklagt unter anderem eine nur oberflächliche Recherche der Pleiades Entwickler, welche zum großen Teil aus klassischen Archäologen bestehen und einen großen Bogen um Gebiete wie Polen und der Slowakei gemacht haben. Zudem sind Orte wie Westerndorf[443] und Pfaffenhofen[444] zu einem Gebiet (Pons Aeni[445]) verschmolzen, wobei es archäologisch betrachtet zwei verschiedene Orte sind. Um diese Dinge zu berichtigen ist es möglich Pleiades Places zu bearbeiten[446]. Weitere Auffälligkeiten sind nachfolgend aufgelistet:

- Cannstatt: Sehr wichtiger Fundort, aber ohne Pleiades-Referenz
- Holt: Sehr wichtiger Fundort, aber ohne Pleiades-Referenz
- Inchtuthil: Sehr wichtiger Fundort, aber ohne Pleiades-Referenz (Römischer Name: Pinnata)
- Kräherwald: Canstatt - Teil von Stuttgart, aber eigenständiger Fundort
- Pfaffenhofen: Pfaffenhofen und Westerndorf liegen dicht nebeneinander, sind jedoch nicht gemeinsam als Pons Aeni zu bezeichnen
- Westerndorf: Pfaffenhofen und Westerndorf liegen dicht nebeneinander, sind jedoch nicht gemeinsam als Pons Aeni zu bezeichnen
- Colchester: nicht korrekt erfasst

[442] vgl. http://web.rgzm.de/no_cache/ueber-uns/team/m/allard_mees.html (abgerufen 04.12.2013)

[443] vgl. http://143.93.114.104/rest/samian/findspots/Westerndorf

[444] vgl. http://143.93.114.104/rest/samian/findspots/Pfaffenhofen

[445] vgl. http://pleiades.stoa.org/places/187516 (abgerufen 04.12.2013)

[446] vgl. http://pleiades.stoa.org/welcome (abgerufen 04.12.2013)

Trotz der zuvor aufgezeigten Probleme war ein Mapping vieler Samian Places zu Pleiades Places möglich, vgl. Anhang H: Liste der Mapping Ergebnisse. Die Ergebnisse des manuellen Mapping werden mit Hilfe eines SQL Skripts (6_fill_matchtable.sql) in die PostGIS Datenbanktabelle pleiadesmatch übertragen und stehen allen Anwendungen zur Verfügung.

```
INSERT INTO pleiadesmatch VALUES ('{FindspotName}', '{PleiadesID}');
```

Eine Verbindung in einer relationalen Datenbanktabelle ist mit der Übertragung der Daten geschehen. Die Nutzung dieser im Semantic Web ist jedoch nur durch eine semantische Modellierung der Beziehungen möglich (vgl. Pelagios-Entry-Point, Kapitel 6.1.6). Hierbei ist folgendes zu beachten:

- Samian Places entsprechen nicht exakt den Pleiades Places
- Samian Places sind präziser und detaillierter als Pleiades Places und sollen mit ihnen nur in Verbindung gebracht werden um die Pelagios Datasets zu erreichen
- Samian Places sollen keine Konkurrenz zu Pleiades sein

Tabelle 7-1 erläutert Möglichkeiten der semantischen Modellierung von Verbindungen:

Tabelle 7-1: Semantische Verbindungsmodellierungen

Vokabular	**Erläuterung**
owl:sameAs[447]	wird verwendet wenn eine Ressource einer anderen entspricht
rdfs:seeAlso[448]	wird verwendet wenn eine Ressource zusätzliche Informationen über die Ressource bieten könnte (Bsp. dbpedia)
skos:mappingRelation[449] skos:closeMatch, skos:exactMatch, skos:broadMatch, skos:narrowMatch, skos:relatedMatch	wird verwendet um zwei SKOS Konzepte mit einander zu verbinden und deren Beziehung zu beschreiben
Open Annotation[450]	wird verwendet um zwei Ressourcen (z.B. Textfragment und geografische Information) mit einander in Beziehung zu stellen

[447] vgl. http://www.w3.org/TR/owl-ref/#sameAs-def (abgerufen 04.12.2013)

[448] vgl. http://www.w3.org/TR/rdf-schema/#ch_seealso (abgerufen 04.12.2013)

[449] vgl. http://www.w3.org/TR/skos-reference/#mapping (abgerufen 04.12.2013)

[450] vgl. http://www.openannotation.org/spec/core (abgerufen 04.12.2013)

Das Prädikat owl:sameAs entspricht nicht den Vorgaben der Verbindung, da beide Ressourcen nicht exakt das gleiche Aussagen. Zudem ist auch keine direkte weitere benötigte Information im Pleiades Place über den Samian Place verfügbar, wodurch das prädikat rdfs:seeAlso ebenfalls nicht in Betracht kommt. Samian Places werden nach Vorbild der Pleiades Places (vgl. Kapitel 6.1.5) modelliert und bestehen so aus einem SKOS Konzept. Durch diesen Aspekt wäre ebenfalls eine Modellierung über eine SKOS Beziehung möglich. Diese wird jedoch auf Grund des schwierig abzuschätzenden Beziehungsstatus nicht in Betracht gezogen.

Pelagios nutzt das Konzept der Annotationen zur Modellierung von Referenzen zu Places. Dies soll die Tatsache unterstreichen, dass ein Verweis auf einen Ort als nicht gesichert angesehen werden kann, sondern lediglich eine Aussage über irgendeine Art von Beziehung untereinander tätigt. Technisch wird das Open Annotation Collaboration[451] RDF Model genutzt um eine Information (annotaion boby) mit einer anderen Information (annotation target) zu verbinden. In Pelagios wird der zu verlinkende Inhalt als annotation target und die Pleiades URI ans annotation body angesehen[452] [GitHub, 2012b].

Betrachtet man den Sachverhalt des Mapping von Samian zu Pleiades Places ist nur eine Information gesichert: Eine Ressource hat eine Beziehung zu einer anderen Ressource. Da diese nicht exakt definiert ist, bildet das Open Annotation Modell eine sinnvolle Möglichkeit zur semantischen Modellierung dieser. Dabei folgt die target/body Vergabe den Vorgaben von Pelagios (target:Findspot; body:PleiadesID). Innerhalb der Pelagios Community ist jedoch in den letzten Wochen eine Diskussion[453],[454] gestartet ob diese Definition von Target und Body richtig ist. Ursprünglich wurde Pleiades Places zur Verknüpfung von Textfragmenten mit einem geografischen Ort entwickelt. Eine Beziehung zwischen zwei Orten kam nicht in Betracht. Eine abschließende Meinung ist (zurzeit) noch nicht vorhanden.

Als Beispiel einer Open-Annotation-Verknüpfung sei der Findspot Langenhain mit dessen Pleiades Place 109100 gezeigt. Hierbei ist zu erkennen, dass weitere Merkmale, nicht nur target:Langenhain und body:109100, sondern auch die Motivation, den Erzeuger der Annotation und dessen Datum wie auch eine Beschreibung vorhanden ist.

---

[451] vgl. http://www.openannotation.org/spec/beta (abgerufen 04.12.2013)

[452] Beispiel: http://uredb.reading.ac.uk/ure/ure.n3 (abgerufen 04.12.2013)

[453] vgl. https://groups.google.com/forum/#!topic/pelagios-project/Lw-zgpYWgqk (abgerufen 04.12.2013)

[454] vgl. https://groups.google.com/forum/#!topic/pelagios-project/7Zy0sgt0EP8 (abgerufen 04.12.2013)

```
#Links to Pleiades
#Prefix
@prefix oa: <http://www.openannotation.or4g/ns/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix pleiadesplace: <http://pleiades.stoa.org/places/> .
@prefix anno: <http://143.93.114.104/rest/samian/annotations#>.
@prefix agent: <http://143.93.114.104/rest/samian/agents#>.
@prefix findspot:
<http://143.93.114.104/rest/samian/findspots/>.

anno:Langenhain-109100 a oa:Annotation;
  oa:motivatedBy oa:linking;
  oa:annotatedBy agent:Mees;
  oa:annotatedAt "Tue Nov 19 22:48:57 CET 2013";
  oa:hasBody pleiadesplace:109100;
  oa:hasTarget findspot:Langenhain;
  dcterms:description "Link made by Heinz/Mees/Thiery.".
agent:Mees a foaf:person;
  foaf:name "Allard Mees".
```

7.4 PelagiosConnectionAPI

Kapitel 7.3 erläutert bereits das Mapping der Samian mit Pleiades Places und die Generierung von semantischen Tripeln mittels Open Annotation, wobei der Body der Findspot und das Target die Pleiades ID, ist. In Kapitel 7.2 wird die Pelagios API erläutert, welche eine Graphensuche im Pelagios Netzwerk ermöglicht. Durch die Kombination dieser beiden Vorgänge in der PelagiosConnectionAPI (PCA) ist es somit möglich, gespeicherten Verlinkungen von Samian zu Pleiades Places in Kombination mit der Pelagios API zur automatischen Generierung neuen Wissens über die Samian Places zu nutzen. Die API kann standalone genutzt werden, wird jedoch auch im Linked Samian Ware Explorer zur Anzeige der Pelagios Verlinkungen eingebunden (vgl. Kapitel 8.2).

Der Apache-Tomcat Container beherbergt das File PelagiosConnectionAPI.war, welche das Herzstück der PCA bildet und über folgende URL erreichbar ist:

http://143.93.114.104/PelagiosConnectionAPI

Die API integriert ein Servlet (PelagiosData) und kann durch mehrere Parameter (Findspot / PleiadesID / type / datasets) aufgerufen werden:

- Findspot: Samian Place, zu dem Pelagios Datasets (URI) gesucht werden sollen
- PleiadesID: Pleiades Place, zu dem Pelagios Datasets (URI) gesucht werden sollen
- type: Ausgabetyp der Datasets (zur Zeit nur xml implementiert)
- datasets: true = Ausgabe der Datasets Namen, false = Ausgabe der Datasets URIs

Die Auflistung zeigt, dass sowohl Pelagios Dataset URIs, sowie auch nur deren Namen entweder von einem Samian oder Pleiades Place via HTTP GET abgerufen werden können. Bei der Anfrage über einen Samian Place wird der Triplestore (vgl. Kapitel 6.3) über ein entsprechendes Mapping befragt. Die Ausgabe erfolgt über eine definierte XML Struktur:

```
<list>
      <link>URI/Name</link>
      <link>URI/Name</link>
      ...
</list>
```

Um die Möglichkeiten der PCA zu verdeutlichen, dienen 4 Use-Cases am Beispiel Langenhain (vgl. auch Abbildung 7-7):

Use Case 1: Pelagios Dataset URIs des Findspot Langenhain

```
http://143.93.114.104/PelagiosConnectionAPI/PelagiosData
?Findspot=Langenhain&type=xml&datasets=false
```

Use Case 2: Pelagios Dataset Namen des Findspot Langenhain

```
http://143.93.114.104/PelagiosConnectionAPI/PelagiosData
?Findspot=Langenhain&type=xml&datasets=true
```

Use Case 3: Pelagios Dataset URIs der PleiadesID 109100 (Langenhain)

```
http://143.93.114.104/PelagiosConnectionAPI/PelagiosData
?PleiadesID=109100&type=xml&datasets=false
```

Use Case 4: Pelagios Dataset Namen des PleiadesID 109100 (Langenhain)

```
http://143.93.114.104/PelagiosConnectionAPI/PelagiosData
?PleiadesID=109100&type=xml&datasets=true
```


```
<list>
  <link>http://pleiades.stoa.org/places/109100</link>
- <link>
    http://pelagios.dme.ait.ac.at/api/places/http%3A%2F%2Fpleiades.stoa.org%2Fplaces%2F109100
  </link>
  <link>http://awmc.unc.edu/api/omnia/162088</link>
  <link>http://www.francia.ahlfeldt.se/places/16051</link>
</list>
```

```
<list>
  <link>AWMC - Ancient World Mapping Center</link>
  <link>Regnum Francorum Online</link>
</list>
```


Abbildung 7-7: PelagiosConnectionAPI (Beispielausgabe)

Den Kern des JAVA Programms bilden ein Servlet (PelagiosData) und drei Java-Klassen (Pelagios, Sesame und PostGIS). Das Servlet regelt den Request und Response, verwaltet als zentrale Stelle die Beschaffung der Pleaides ID (ggf. durch Paramater oder den Triplestore) und generiert den Output als application/xml. Die Klasse Sesame liest mittels SPARQL Abfrage die Pleiades ID aus dem Triplestore. Die Verbindung mit der Pelagios API und der Erhalt der Datasets werden über die Klasse Pelagios gesteuert. Die Klasse PostGIS stellt eine Verbindung zur Tabelle pleiadesmatch her, wird jedoch nicht weiter genutzt[455]. Mit Hilfe der Maven Dependency postgresql[456] wird der Treiber zur Ansprache der PostGIS Datenbank eingebunden. Die Erstellung eines RDF Graphen und dessen Befragung in JAVA ist mit der Bibliothek Apache Jena[457] möglich. Das Parsen von JSON-Objekten übernimmt die Bibliothek json-simple[458]. Der Programmablauf (hier Use Case 1) der PCA ist Abbildung 7-8 zu entnehmen.

[455] Die Kommunikation erfolgt nur über den Triplestore, ist jedoch auch über die PostGIS möglich

[456] http://mvnrepository.com/artifact/postgresql/postgresql/9.1-901.jdbc3 (abgerufen 04.12.2013)

[457] http://mvnrepository.com/artifact/org.apache.jena/jena-arq/2.11.0 (abgerufen 04.12.2013)

[458] http://mvnrepository.com/artifact/com.googlecode.json-simple/json-simple/1.1.1 (abgerufen 04.12.2013)

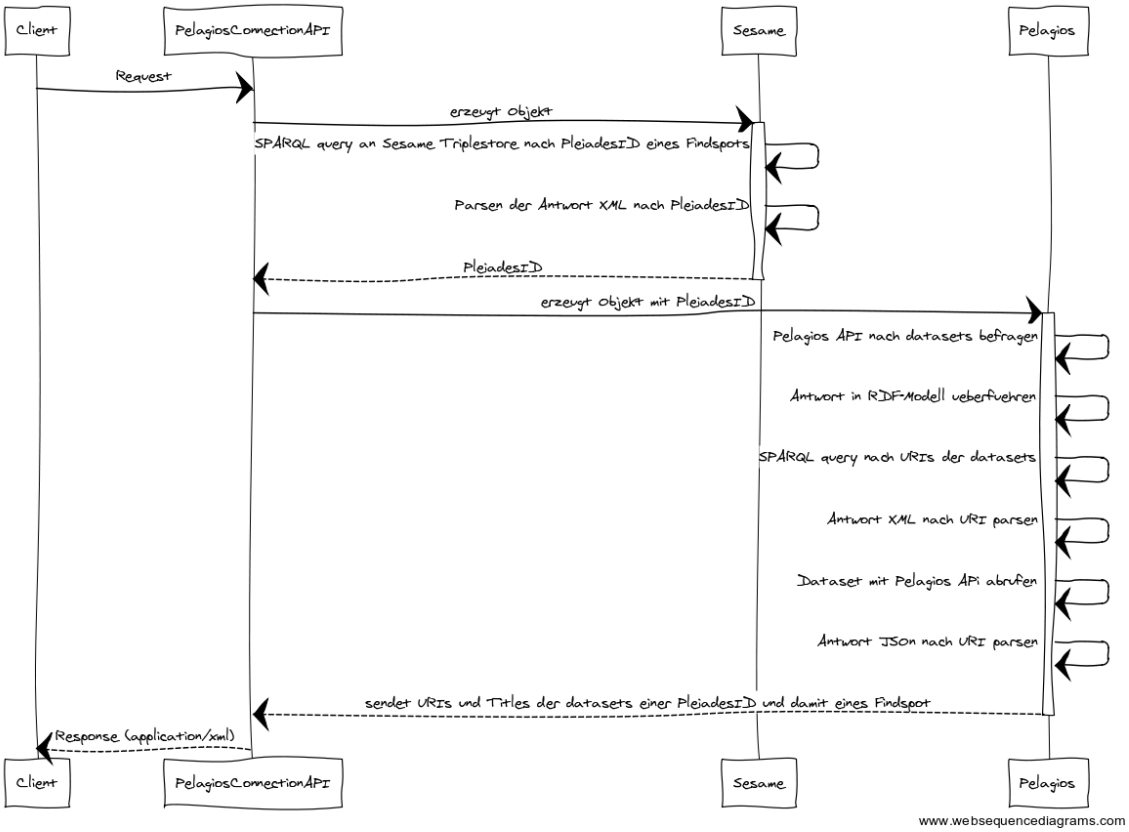


Abbildung 7-8: PelagiosConnectionAPI (Sequenzdiagramm)

Der Request des Clients stößt das Parsen der Eingabeparameter des Servlets und die Erzeugung einer Instanz der Sesame Klasse an. Diese bereitet eine SPARQL-Query an den Triplestore zum Erhalt der PleiadesID des Findspots vor:

```
SELECT DISTINCT ?erg WHERE {
      ?a oa:hasBody ?erg .
      FILTER(REGEX(STR(?erg),
      \"^http://pleiades.stoa.org/places/\"))
      ?a oa:hasTarget findspot:{findspot} .
      }
```

Das Ergebnis, als SPARQL Query Results XML[459] (SQR XML), wird mit SaxBuilder[460] geparst und die Pleiades ID an das Servlet übergeben. Eine Instanz der Pelagios Klasse erhält die Datasets der

[459] vgl. http://www.w3.org/2005/sparql-results# und http://www.w3.org/TR/rdf-sparql-XMLres (04.12.2013)

[460] http://www.jdom.org/docs/apidocs/org/jdom2/input/SAXBuilder.html (abgerufen 04.12.2013)

PleiadesID (vgl. Kapitel 7.2) als RDF/XML und befragt[461] diese mittels Apache Jena[462] via SPARQL nach den einzelnen Datasets:

```
SELECT ?dataset WHERE {
      ?dataset a void:Dataset
}
```

Die Abfrage ergibt ein weiteres SQR XML, welches die URIs der Pelagios Datasets enthält. Ein weiteres Parsen dieser XML stellt die Grundlange der Anfrage an die Pelagios API nach den Annotationen der einzelnen Datasets im JSON Format dar (vgl. Kapitel 6.3). Das enthaltene JSON-Objekt wird nun nach target, body und title durchsucht und beinhaltet somit die URIs und Namen der mit dem Findspot verlinkten Ressource, welche an das Servlet zurückgegeben werden. Als Ausgabe wird ein application/json generiert.

Die PelagiosConnectionAPI stellt somit eine direkte Verbindung zwischen Samian Places (Findspots) und an Pelagios Projekt angeschlossenen Ressourcen und Datasets her. Dies bietet den großen Vorteil, dass automatisch neues Wissen (sobald neue Datasets in Pelagios eingepflegt werden) zu Findspots ermittelt werden kann, ohne manuelle Einstellungen vorzunehmen. Im Oktober 2013 wurde z.B. ein neuer Datensatz der American Numismatic Society[463] mit Pelagios verknüpft, sodass neue Münzfunde als Ressourcen für einen Findspot direkt verfügbar sind[464].

7.5 Beurteilung

Das Mapping von Samian Places und Pleiades Places kann als problematisch gesehen werden (falsche Modellierungen, andere Bedeutungen, etc.). Zudem ist das manuelle Mapping mit einem enormen Arbeits- und Zeitaufwand verbunden. Die Verbindung beider Placetypen wird mit Hilfe von Open Annotation gelöst. Dieses von Pelagios übernommene Verfahren ist jedoch ursprünglich zur Annotation von Textfragmenten und Pleiades Ortsressourcen gedacht. Sollte eine anders geartete Verknüpfung zwischen zwei Ortstypen entwickelt werden, ist diese primär zu verwenden. Das Mapping zu Pleiades und die Verwendung der Pelagios API ermöglicht den automatischen Erhalt, neuer an das Pelagios Projekt angeschlossener Datasets. Durch dieses Verfahren können die eigenen Samian Places mit wertvollen, in den Kontext einschließbaren, Informationen angereichert werden.

---

[461] http://www.ibm.com/developerworks/xml/library/j-sparql/ (abgerufen 04.12.2013)

[462] http://jena.apache.org/tutorials/ (abgerufen 04.12.2013)

[463] http://pelagios.dme.ait.ac.at/api/datasets/95f6ab003ed45c4aba5b153cf600274e (abgerufen 04.12.2013)

[464] z.B. http://143.93.114.104/PelagiosConnectionAPI/PelagiosData?Findspot=Colchester&type=xml&datasets=false (http://numismatics.org/collection/1979.22.1)

8 Die App - Prototypische Webanwendung

Die Zusammenführung der in Kapitel 6 (Datenbereitstellung) und Kapitel 7 (Datenverknüpfung) vorgestellten Dienste und Funktionen erfolgt in einer prototypisierten Webanwendung, der App[465]. Die Hauptseite der Applikation ist mit der URL http://143.93.114.104/app zu erreichen.

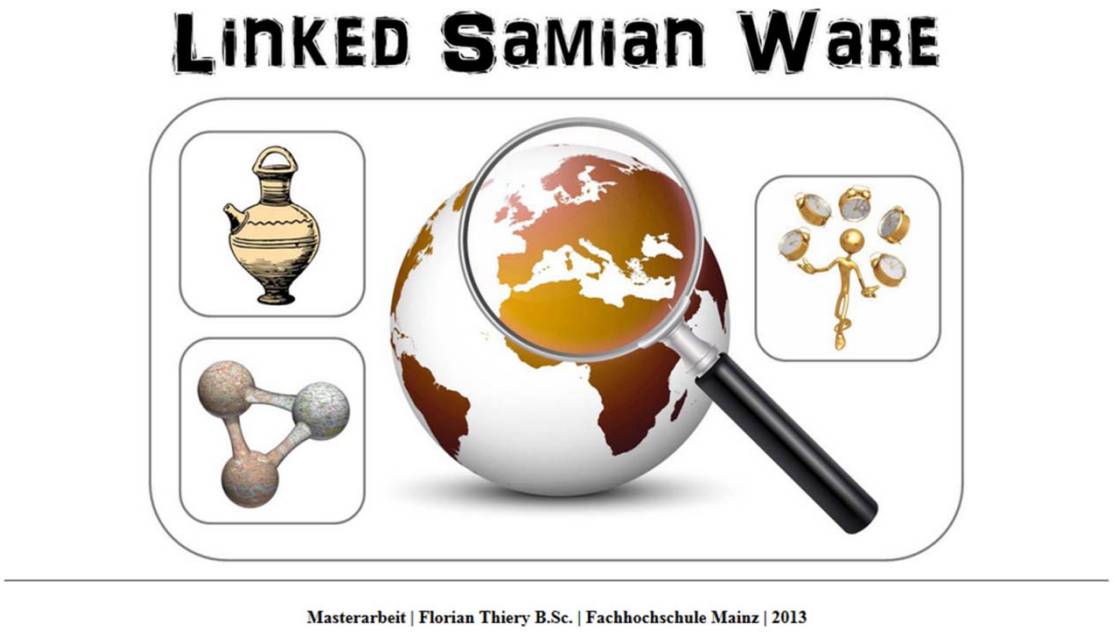


Abbildung 8-1: Die App

Abbildung 8-1 zeigt die vier Komponenten der App:

- ReST-Schnittstelle (vgl. Kapitel 6.1; links oben)
- Linked Samian Ware Explorer (LSWE; links unten)
- Kartenansicht der Findspots (Map; Mitte)
- Linked Time Explorer (LTE, rechts)

[465] siehe auch http://de.wikipedia.org/wiki/Anwendungssoftware (abgerufen 04.12.2013)

Jede Komponente ist mit einer eigenen URL aufrufbar:

- ReST: http://143.93.114.104/rest
- LSWE: http://143.93.114.104/explorer
- Map: http://143.93.114.104/map
- Time Explorer: http://143.93.114.104/timexplorer

Die nachfolgenden Kapitel erläutern die Möglichkeiten und die Funktionsweise des LSWE und der Map. Der Quellcode ist Anhang J: Quellcode der Apps zu entnehmen.

8.1 Die Map - Kartenansicht der Findspots

Die in der Datenbank enthaltenen Geometrien der Findspots als WGS84-Koordinaten (vgl. Kapitel 5) und die Bereitstellung von OCG-konformen Webdiensten wie WFS und WMS (vgl. Kapitel 6.2) bieten eine Vielzahl von webbasierten Karten-Visualisierungsmöglichkeiten. Beispiele wie Regnum Francorum Online[466] oder auch ORBIS[467], welches interaktiv Routen auf dem Land und Wasser durch das antike römische Reich visualisiert, zeigen das Potential von Kartenanwendungen in den Geisteswissenschaften.

Die webbasierte Kartenansicht kann mit einer Vielzahl von bereits existierenden, freien und erweiterbaren Anwendungen erfolgen. Erwähnt seien hier die JavaScript basierten Bibliotheken Leaflet[468] und Openlayers[469]. Leaflet ist eine open-source JavaScript Bibliothek, die besonders gut für mobile Anwendungen geeignet ist. Sie zeichnet sich besonders durch ihre kleine Größe (34KB), Einfachheit, Benutzerfreundlichkeit und Leistungsfähigkeit aus. Die Bibliothek bietet eine Vielzahl an Funktionen[470] und Plug-Ins[471] an und nutzt HTML5 und CSS3 Elemente [LEAFLET, 2013A]. Open Layers bietet ähnliche Funktionalitäten an und basiert ebenfalls auf JavaScript, sowie HTML5 und CSS3 Elementen [OPENLAYERS, 2013]. Vergleich man beide Bibliotheken, ist besonders die größere Anzahl der Entwickler und geringere Anzahl an Sourcecode bei Leaflet[472], aber auch die Unterstützung des Internet Explorers von Open Layers[473] zu erwähnen. Auf Grund der einfachen Struktur des Quellcodes und der Vielzahl an Funktionen und Plugins, wird für alle Kartenanwendungen die Bibliothek Leaflet verwendet.

---

[466] vgl. http://www.francia.ahlfeldt.se (abgerufen 04.12.2013)

[467] vgl. http://orbis.stanford.edu (abgerufen 04.12.2013)

[468] vgl. http://leafletjs.com (abgerufen 04.12.2013)

[469] vgl. http://openlayers.org (abgerufen 04.12.2013)

[470] vgl. http://leafletjs.com/features.html (abgerufen 04.12.2013)

[471] vgl. http://leafletjs.com/plugins.html (abgerufen 04.12.2013)

[472] vgl. http://www.ohloh.net/p/compare?project_0=OpenLayers&project_1=Leaflet (abgerufen 04.12.2013)

[473] vgl. http://lydonchandra.com/content/comparison-openlayers-vs-leaflets (abgerufen 04.12.2013)

Leaflet unterstützt die Einbindung von WMS- und WFS-Diensten, welche in allen Anwendungen mit Hilfe des Geoservers geschieht. Als WMS werden PNG-Bilder[474] eingebunden, das WFS-Austauschformat ist GeoJSON [LEAFLET, 2013B]. Abbildung 8-2 zeigt eine Leaflet Grundkarte (Basemap[475]) und die Einbindung eines WFS (Findspots) als Layer. Kapitel 6.2 beschreibt Probleme bei der direkten Einbindung des GeoJSON-Response des Geoservers, welche durch die Verwendung eines Proxy[476] behoben werden. Des Weiteren ist kein eigenes Plug-In zur Einbindung eines WFS vorhanden. Die App getFeature[477]stellt den zuvor erwähnten Proxy und ein Plug-In zur Einbindung eines WFS dar und sichert die Bereitstellung von GeoJSON Objekten in Leaflet-konformen Formaten.

Abbildung 8-2: Leaflet Karte mit Findspots

Die Map zeigt neben einer Vielzahl von verschiedenen Baselayern (aktike Layer wie Imperium[478] oder moderne Layer wie Cloudmade Fresh[479]) und eigenen Findspotlayern der Samian Datenbank eine Vielzahl von nützlichen Plug-Ins[480]: eine Minimap[481] (zur Anzeige einer kleinen Übersichts-

---

474 z.B. Samian:places (abgerufen 04.12.2013): http://143.93.114.104/geoserver/Samian/wms?service=WMS&version=1.1.0&request=GetMap&layers=Samian:places&styles=&bbox=-4.311666,0.0,19.083334,56.541348&width=211&height=512&srs=EPSG:4326&format=image%2Fpng

475 weitere Basemaps: https://github.com/leaflet-extras/leaflet-providers (abgerufen 04.12.2013)

476 siehe auch http://de.wikipedia.org/wiki/Proxy_(Rechnernetz) (abgerufen 04.12.2013)

477 z.B. http://143.93.114.104/getfeature/findspots?param=all

478 vgl. http://pelagios.dme.ait.ac.at/maps/greco-roman (abgerufen 04.12.2013)

479 vgl. http://cloudmade.com (abgerufen 04.12.2013)

480 vgl. http://leafletjs.com/plugins.html (abgerufen 04.12.2013)

karte), einen LayerSwitcher[482] (zum Wechseln zwischen Verschiedenen Base- und Findspotlayern), einem Vollbildmodus[483] (zur Ansicht auf dem kompletten Bildschirm) und verschiedene MarkerIcons[484] (zur Darstellung verschiedener Sachverhalte, wie isKilnsite oder noKilnsite). Diese Plug-Ins sind in Abbildung 8-3 durch rote Rahmen beschrieben.

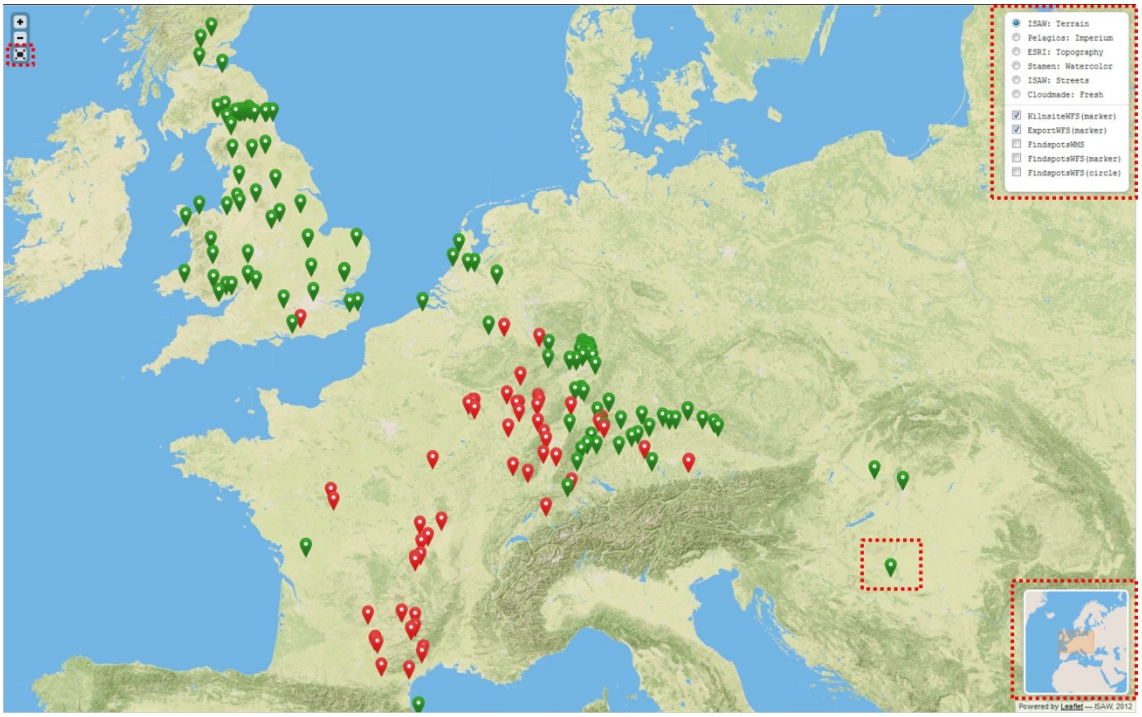


Abbildung 8-3: Leaflet Plug-Ins der Map

8.2 Der Linked Samian Ware Explorer (LSWE)

Der Linked Samian Ware Explorer (LSWE) dient zum einen der Anzeige der Beziehungen innerhalb der 'Samian-Ware-Datenbank', zum Anderen als Verknüpfungs- und Visualisierungstool zwischen den Findspots und den Pleides Places und damit den an PELAGIOS angeschlossenen Daten.

---

[481] vgl. https://github.com/Norkart/Leaflet-MiniMap (abgerufen 04.12.2013)

[482] vgl. http://leafletjs.com/examples/layers-control.html (abgerufen 04.12.2013)

[483] vgl. http://brunob.github.io/leaflet.fullscreen (abgerufen 04.12.2013)

[484] vgl. http://leafletjs.com/examples/custom-icons.html (abgerufen 04.12.2013)

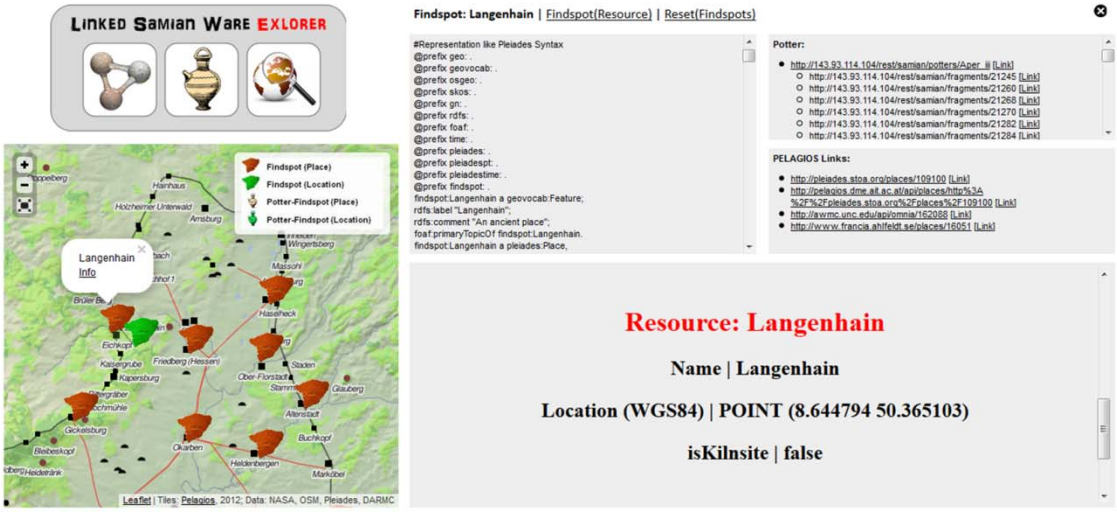


Abbildung 8-4: Explorer (Langenhain)

Abbildung 8-4 gibt einen Überblick über die im Linked Samian Ware Explorer enthaltenen Funktionen. Der Explorer ist eine JavaScript Anwendung[485] mit Unterstützung der jQuery- und Leaflet-Bibliothek und HTML5 bzw. CSS3 Elementen[486]. Die Leaflet-Karte[487] auf der rechten Seite zeigt alle Findspots mit dem Symbol einer braunen Scherbe (Locations werden grün dargestellt), als Kartengrundlage dient die PELAGIOS Imperium Basemap[488]. Diese Findspots werden, wie bereits in Kapitel 8.1 beschrieben, mittels eines Proxy über den Geoserver als GeoJSON eingebunden[489]. In Bereichen, in denen sich Fundorte häufen und die Differenzierung der einzelnen Orte nicht mehr gegeben ist, wird sich der Technik des Clustern bedient, d.h. dass diese Bereiche zu einem Gebiet zusammengefasst und als Kreis mit der Anzahl der Findspots angezeigt werden. Zum Clustern dient das Leaflet-Plugin[490] Leaflet.markercluster[491] (Dave Leaver), das je nach Zoomlevel das Clusterverhalten verändert.

Nach Einbindung der erforderlichen Dateien, der Erstellung einer neuen MarkerClusterGroup, dem Hinzufügen der einzelnen Marker zu jener Gruppe und dem Hinzufügen dieser zur Map ist die Grundeinstellung zum Clustern abgeschlossen[492].

---

[485] Der Quellcode kann der explorer/index.htm und explorer/linkfunctions.js entnommen werden

[486] Quellcode vgl. Anhang J: Quellcode der Apps

[487] Informationen zu Leaflet und der Programmierung sind Kapitel 8.1 zu entnehmen

[488] vgl. http://pelagios.dme.ait.ac.at/tilesets/imperium//{z}/{x}/{y}.png

[489] Einbindung über den Proxy: http://http://143.93.114.104/getfeature/findspots?param=all

[490] vgl. http://leafletjs.com/plugins.html (abgerufen 04.12.2013)

[491] vgl. https://github.com/Leaflet/Leaflet.markercluster (abgerufen 04.12.2013). Informationen zur Benutzung, den Einstellungen, und die benötigten Dateien können hier von Github abgerufen werden.

[492] vgl. https://github.com/Leaflet/Leaflet.markercluster (abgerufen 04.12.2013)

```
<script type="text/javascript" src="leaflet.markercluster-src.js"></script>
<link rel="stylesheet" href="MarkerCluster.css" />
<link rel="stylesheet" href="MarkerCluster.Default.css" />

var markers = new L.MarkerClusterGroup();
markers.addLayer(new L.Marker(getRandomLatLng(map)));
... Add more layers ...
map.addLayer(markers);
```

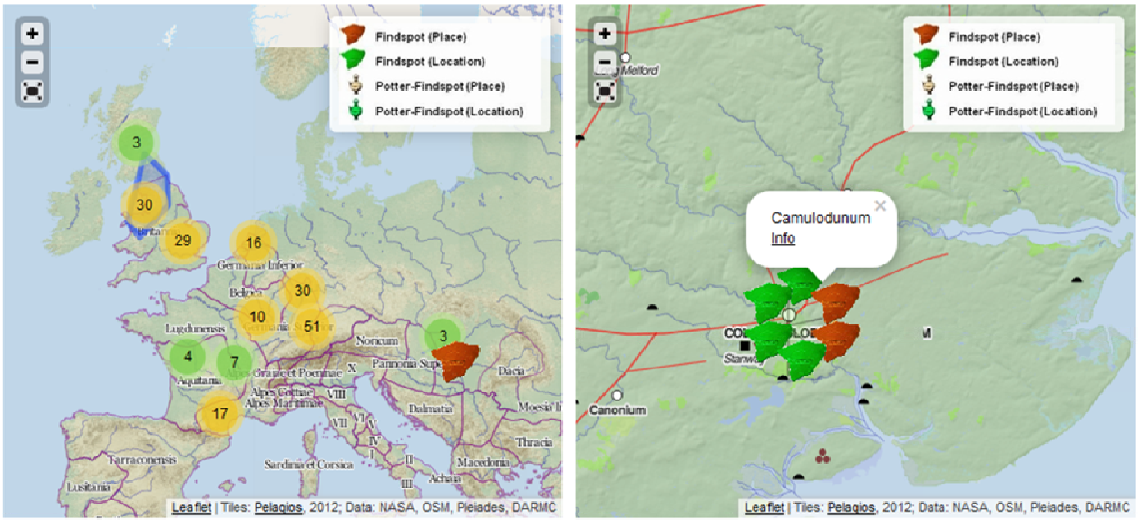


Abbildung 8-5: Leaflet Clustering

In Abbildung 8-5 (rechts) ist erkennen, dass größere Gebiete (z.B. Zentralbrittannien) zusammengefasst werden. Das blaue Polygon gibt den Umring an, in welchem in diesem Fall, 30 weitere Fundorte vorhanden sind. Die rechte Seite zeigt das Beispiel Colchester (Südwestengland). Der Place Colchester besitzt einige Locations, die mit der gleichen Koordinate versehen sind. In der höchsten Zoomstufe ist es mit Hilfe des Cluster Plug-Ins möglich, die einzelnen Locations aufzulösen und mit einer 'Spider-Ansicht' zu versehen.

Die Eigenschaften des Clustern können in leaflet.markercluster-src.js manipuliert werden. Der maximale Clusterradius (maxClusterRadius) gibt den größtmöglichen Radius in Pixeln an, der von einem zentralen Marker gruppiert wird. In dieser Masterarbeit wird der Radius auf 35px gesetzt, welches der Höhe der Scherbengrafik entspricht. Somit ist eine Vermeidung der Überlappung von Grafiken sichergestellt. Die Eigenschaft spiderfyOnMaxZoom = true dient zur Erstellung der Spideransicht[493] (vgl. Abbildung 8-5, rechts).

Mit Klick auf eines der Scherbensymbole öffnet sich ein Popupfenster, welches den Namen des Findspots anzeigt und dem Nutzer die Möglichkeit bietet, weitere Informationen (z.B. Verlinkungen) des Findspots zu erhalten (vgl. Abbildung 8-4, rechts).

[493] Weitere Einstellmöglichkeiten: https://github.com/Leaflet/Leaflet.markercluster (abgerufen 04.12.2013)

Die Informationsanzeige gliedert sich in vier Bereiche:

- Ansicht des Findspots in RDF/Turtle Syntax (oben links)
- Ansicht der Verknüpfungen zu Potter-Ressourcen (und deren gestempelten Fragmente), sowie zu Fragment-Ressourcen (und zu jenen, die sie gestempelt haben) des Findspots (oben rechts)
- Ansicht der Verknüpfungen zu PELAGIOS, welche mit der gleichen Pleiades ID gekennzeichnet sind (mitte rechts)
- IFrame zur Ansicht verschiedener Ressourcen (unten)

Die Repräsentation des Findspots in RDF-Syntax entspricht der Abfrage der ReST-Schnittstelle (hier: http://143.93.114.104/rest/samian/findspots/Langenhain.ttl). Zur Anzeige der Verlinkungen innerhalb des Samian-Datensatzes, sowie derer zu PELAGIOS, wird zur Vereinfachung des Auslesens in JavaScript in der ReST-Schnittstelle der Postfix {ressource}.exp, und damit eine XML-Schnittstelle, der als RDF vorliegenden Tripel, eingeführt. Findspots sind somit über die URL http://143.93.114.104/rest/samian/{ressourses}/{ressource}.exp in einer XML-Ansicht in folgender Struktur verfügbar und werden zur Ansicht in HTML geparst:

```
<list>
      <potter uri="...">
            <fragment_p><uri>...</uri><uri>...</uri><fragment_p>
      </potter>
      …
      <fragment uri="...">
            <potter_f><uri>...</uri></potter_f>
      </fragment>
</list>
```

Wie bereits in Kapitel 7.4 beschrieben, bietet die PelagiosConnectionAPI ebenfalls eine Ausgabe in XML[494] mit verwandtem Schema an und kann ähnlich in JavaScript behandelt werden. Ein Klick auf einen der Links erzeugt die Ansicht der entsprechenden Ressource im iFrame, im unteren Bereich des Explorers. Informationen eines Potters werden jedoch nicht nur als Ressource im iFrame angezeigt, vgl. Abbildung 8-6.

[494] z.B. http://143.93.114.104/PelagiosConnectionAPI/PelagiosData?Findspot=Aardenburg&type=xml

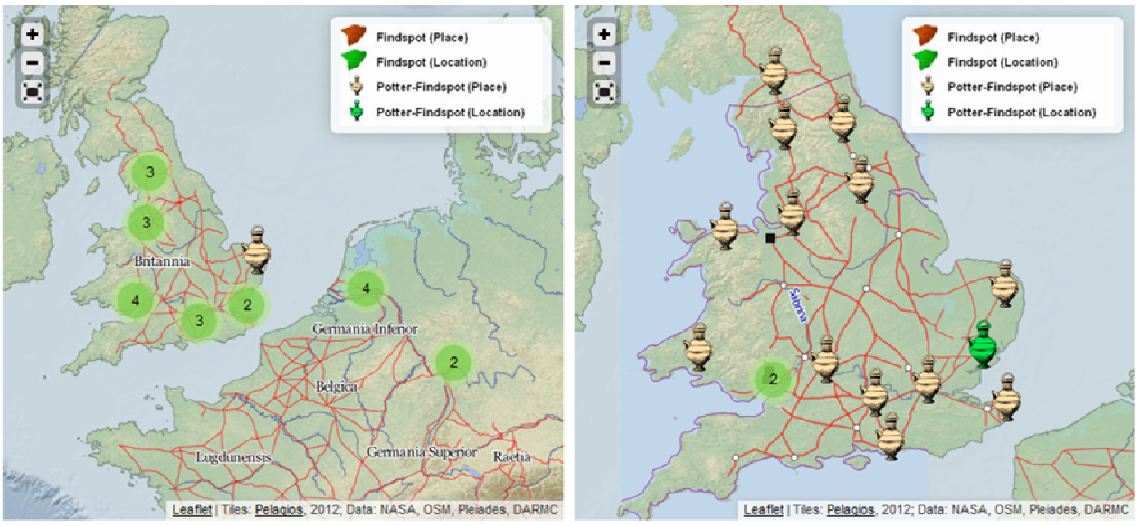


Abbildung 8-6: Findspots eines Potters (Balbinus)

Mit Hilfe des Postfix {potter}.exp in der ReST-Schnittstelle kann einem Potter ein GeoJSON zugeordnet werden[495]. Dieses beinhaltet die Koordinaten und Attribute der Findspots[496], in denen ein Potter durch ein Fragment nachgewiesen wurde[497]. Die Findspots des Potters werden zur Unterscheidung der allgemeinen Fundorte mit einer Karaffe dargestellt. Diese Marker ersetzen den ursprünglichen Layer der Findspots, haben jedoch die gleichen Eigenschaften (Popup, Clustering). Der Link Reset (Findspots) stellt den Ursprungszustand mit der Anzeige aller Findspots her.

[495] z.B. http://143.93.114.104/rest/samian/potters/Balbinus.exp

[496] analog der Ausgabe des Geoservers: http://http://143.93.114.104/getfeature/findspots?param=all

[497] analog der RDF-Ausgabe: http://143.93.114.104/rest/samian/potters/Balbinus.ttl

9 Relative Chronologie

Fast jedes Informationssystem in den historischen und kulturwissenschaftlichen Disziplinen ist auf einen mehr oder weniger genauen Datierungsmechanismus angewiesen. Der Bedarf der maschinellen Ansprechbarkeit von historisch relevanten Zeiträumen und deren Anordnung zueinander, in standardisierter Form, wurde bereits in verschiedenen Projekten formuliert. Als Resultat haben sich feingegliederte und strukturierte Vokabulare entwickelt, die zeitliche Zusammenhänge ausdrücken können. So ist die Möglichkeit der Darstellung von Epochennamen zu exakten Datierungen in STAR oder CLAROS, zur Verbesserung des Retrievals, gegeben. Datierungskomponenten werden aber zumeist systemspezifisch geplant und implementiert. Die Semantic Web Community stellt Zeitkonzepte wie XML Schema[498], OWL Time[499] und TimeML[500], insbesondere für moderne Zeiträume, zur Verfügung. HOLMEN & ORE (2009) plädieren für die Modellierung historischer Gegebenheiten als so genannte Events, wie Produktion und Modifikation. Diese können zudem in größeren Zeiträumen (Epochen) enthalten sein. Der Ansatz des Events wird insbesondere in CIDOC CRM angewendet. Ein Vergleich verschiedener Formen der Ereignismodellierung ist SHAW ET AL. (2009) zu entnehmen. Darunter fällt auch der Ansatz von ALLEN (1983), dessen Methode der temporalen Intervall-Logik von einem absolut chronologischen Ansatz zu einem relativchronologischen Ansatz übergeht. CIDOC CRM nutzt ebenfalls den Ansatz nach Allen (vgl. HOLMEN & ORE, 2009 und Abbildung 9-1). Eine Bereitstellung der Datumsinformationen in relativ chronologischen Zeiträumen, die in einer Beziehung zueinander stehen, würde zu einer Verbesserung der Interoperabilität beitragen. Bezüge könnten um eine weitere Dimension, die Zeit, angereichert werden und somit weitreichende Perspektiven für kulturwissenschaftliche Beziehungen erzeugen.[501]

---

[498] siehe auch http://www.w3.org/TR/xmlschema-2/#built-in-primitive-datatypes (abgerufen 04.12.2013)

[499] siehe auch http://www.w3.org/TR/owl-time (abgerufen 04.12.2013)

[500] siehe auch http://timeml.org/site/index.html (abgerufen 04.12.2013)

[501] Information aus einem Antragschreiben des DAI und i3mainz, Stichwort: chronOntology

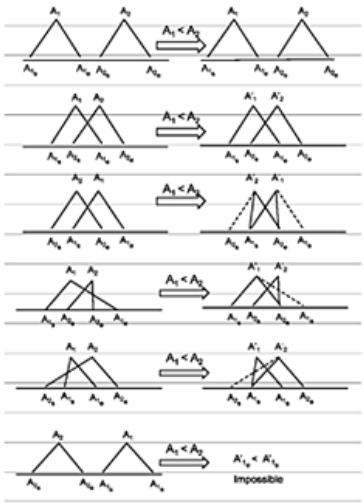


Abbildung 9-1: Temporale Logik nach CIDOC CRM [KGEO, 2013A]

Neben der in Kapitel 6 beschriebenen Bereitstellung der Attribut- und Geometriedaten, ist daher ebenfalls eine Veröffentlichung der in den Daten enthaltenen Datierungen als Linked Data wünschenswert. Die in dieser Arbeit genutzte PostGIS Datenbank enthält bereits in der Tabelle Findspot zwei Datumsfelder: datemin und datemax. Die Datumsfelder enthalten absoultchronologische Angaben, z.B. mindate = 10 und maxdate = 65 (Camulodunum). Diese Angaben sind jedoch mit einigen Ungenauigkeiten und implizitem Wissen (die Datierung des Zeitraums Camulodunums hängt beispielsweise auch von den zu dieser Zeit entwickelten Gefäßformen ab) behaftet. Ein relativchronologischer Ansatz mittels RDF ist somit absoluten Datumsangaben vorzuziehen, da dieser Änderungen in der Chronologie besser abbilden kann und den historischen Hintergrund besser widerspiegelt. Verschiebt sich eine Zeitperiode im Vergleich zu einer anderen, hat dies massive Auswirkungen auf alle anderen chronologischen Bezüge. Diese Auswirkungen sind eindeutig in einem RDF-Graphen ablesbar. Beschreibt man diese Bezüge als absolute Zahlen, bewirkt eine Änderung einer Zeitperiode nicht einer Änderung in der Gesamtstruktur.

Die Möglichkeit der Darstellung einer zeitlich topologischen Beziehung (auch 'Topotime') nach ALLEN (1983) und FREKSA (1992) in relationalen Datenbanksystem wie PostgreSQL oder in RDFs- bzw. OWL- Repräsentationen ist auf Grund der zuvor gezeigten Vorteile verlockend. Relationale Datenbanken lassen es zurzeit zu (siehe oben), Datenfelder nach ISO 8601 mit absolut chronologischen Angaben zu füllen (z.B. Datum oder Jahr oder Stand- und Endfelder in der Form yyyy-mm-dd bzw. nnnn). Mit Hilfe der Operatoren <, > und = können die 13 Relationen der Allen Intervall Algebra (vgl. Kapitel 9.1) berechnet werden. Wie schon zuvor erwähnt, ist diese Algebra in der Welt des Semantic Web bereits in CIDOC CRM integriert[502]. Welche weiteren Ansprüche die Geisteswissenschaften an das Modell stellen, ist einer Diskussion eines Panel[503] (Issues in Spatio-Temporal Technologies for the Humanities and Arts) der Digital Humanities Konferenz in Loncoln,

[502] siehe auch https://cga-download.hmdc.harvard.edu/publish_web/SpaceTime/DH2013_Ore_EventChronology.pdf (04.12.2013)

[503] vgl. http://www.fas.harvard.edu/~chgis/gazetteer/dh_2013.html (abgerufen 04.12.2013)

2013, zu entnehmen. Eine Erweiterung für PostgeSQL (PostgeSQL-Temporal[504]) stellt beispielsweise SQL Funktionen zur Berechnung von Beziehungen von Zeitintervallen zur Verfügung. Eine weitere Ergänzung dieser Bibliothek um Freksas temporal conceptual neighbourhoods (Freksas konzeptionelle Zeitnachbarschaften), welche auf den 13 Beziehungen Allens aufbauen, zeigt GROSSNER (2013A). Zeitperioden können jedoch auch als nicht starr angesehen werden. Start- und Endzeitpunkt können unklar definiert sein (z.B. wahrscheinlich in späten Sommer 2013 oder im August 2013). Eine Lösung für diese unscharfen (fuzzy) Daten bieten beispielsweise die Mitglieder der FinnOnto Group[505] (vgl. KAUPINNEN ET AL., 2010) an. Diese, wie auch weitere Ausführungen, sind GROSSNER (2013B) zu entnehmen.

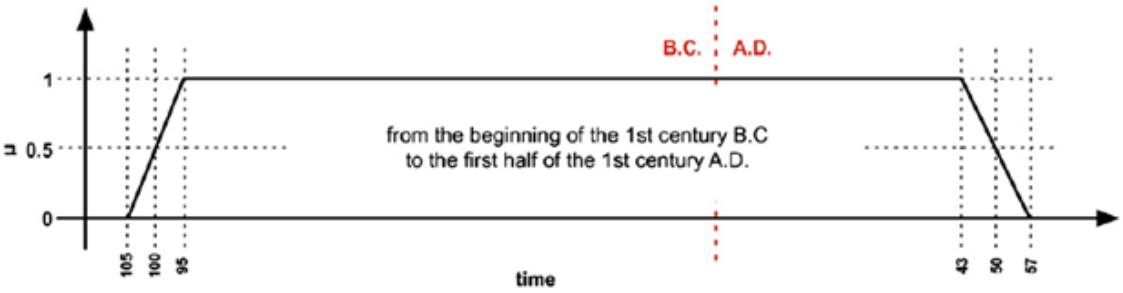


Abbildung 9-2: Fuzzy Times der FinnOntoGroup [KGEO, 2013B]

Die relative Chronologie (abseits von CIDOC CRM) wird daher mit Hilfe des Ansatzes nach Allen und Freksa, da alle Relationen und Schlussfolgerungen aus dem entstandenen Modell abgeleitet werden können, verfolgt (vgl. Kapitel 9.1), in einer prototypische Ontologie umgewandelt (vgl. Kapitel 9.2) und in einer Beispielanwendung, dem Time Explorer (vgl. Kapitel 9.4), mit geeigneten Visualisierungsstrategien (vgl. Kapitel 9.3) exemplarisch umgesetzt.

9.1 Temporale Logik nach Allen und Freksa

Die Intervallalgebra nach Allen (oder auch das Allen-Kalkül[506]) beschreibt eine Logik zur Repräsentation von zeitlichen Zusammenhängen. Zwischen diesen Intervallen können mit Hilfe eines Algorithmus und Beschreibungen von zeitlichen Ereignissen, Schlüsse gezogen werden. ALLEN (1983) stellt 13 verschiedene Relationen (vgl. Abbildung 9-3) dar, die alle möglichen Zusammenhänge zwischen zwei Intervallen aufzeigen.

[504] vgl. https://github.com/jeff-davis/PostgreSQL-Temporal (abgerufen 04.12.2013)

[505] siehe auch http://www.seco.tkk.fi/projects/finnonto/ (abgerufen 04.12.2013)

[506] siehe auch http://de.wikipedia.org/wiki/Allen-Kalk%C3%BCl (abgerufen 04.12.2013)

Temporal Relation	Basic Symbol	Inverse Symbol	Pictorial Example
X *before* Y	<	>	XXX YYY
X *equal* Y	=	=	XXX YYY
X *meets* Y	m	mi	XXXYYY
X *overlaps* Y	o	oi	XXX YYY
X *during* Y	d	di	XXX YYYYYY
X *starts* Y	s	si	XXX YYYYY
X *finishes* Y	f	fi	XXX YYYYY

Abbildung 9-3: Allen Relationen [ALLEN, 1983]

Diese 13 Relationen sind: before (<), after (>), during (d), contains (di), overlaps (o), overlapped-by (oi), meets (m), met-by (mi), starts (s), started-by (si), finishes (f), finished-by (fi), und equals (=). Zur besseren Verdeutlichung können diese Relationen grafisch dargestellt werden (vgl. Abbildung 9-4).

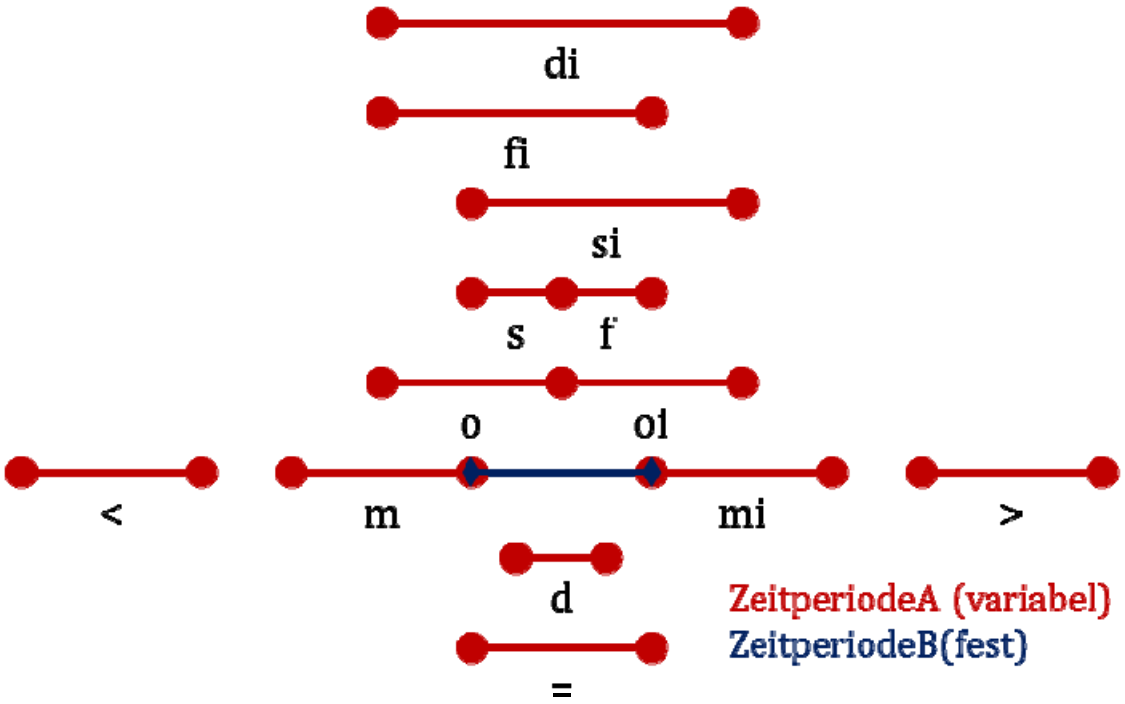


Abbildung 9-4: Allen Relationen (eigene Darstellung)

Die Relationen werden wie folgt gelesen: ZeitpunktA(rot) < Zeitpunkt B(blau). Das heißt zum Beispiel, dass Periode A vor Zeitpunkt B liegt. Tabelle 9-1 gibt näheren Aufschluss über die Relationen und deren Bedeutungen (ein i steht für die Inverse der zuvor erwähnten Relation):

Tabelle 9-1: Allen Relationen (Beispiele)

Beispiel	**Relation**	**Erklärung**
A = B	equals (=)	A ist gleich B
A < B	before (<)	A ist vor B
A > B	after (>)	A ist nach B
A d B	during (d)	A ist während B
A di B	contains (di)	A beinhaltet B
A o B	overlaps (o)	A startet vor und endet in B
A oi B	overlapped-by (oi)	A startet in und endet nach B
A m B	meets (m)	A startet vor und endet mit B
A mi B	met-by (mi)	A startet mit und endet nach B
A s B	starts (s)	A startet mit und endet in B
A si B	started-by (si)	A startet mit und endet nach B
A f B	finishes (f)	A startet in und endet mit B
A fi B	finishes-by (fi)	A startet vor und endet mit B

Aus diesen Relationen können nach ALLEN (1983) intervall-basierte Rückschlüsse gezogen werden. FREKSA (1991) zeigt hierzu zwei kleine Beispiele.

Beispiel 1:

Geht ein Event A unmittelbar Event B voraus (A meets B) und Event C findet während Event B statt (C during B), so kann davon ausgegangen werden, dass Event A vor Event C stattgefunden hat:

```
AAAAABBBB
    CC
```

Beispiel 2:

Findet ein Event A während Event B (A during B) statt und ein Event C startet mit B (C starts B), so kann erreicht werden, dass A entweder during, finishes, overlapped-by, met-by oder after C ist. Keine der acht übrigen Relationen kann zwischen A und C bestehen:

```
  AA
BBBBBBBBB
C
CC
CCC
CCCC
CCCCC
```

Alle Schlussfolgerungen der 13 Relationen (vgl. ALLEN, 1983) und der Bedingung A r1 B; B r2 C können einer Gesamttabelle (siehe Anhang K: Allen und Freksa Schlussfolgerungen) entnommen werden. Das Nachschauen in dieser Tabelle ist eine effiziente menschliche Methode, zeigt jedoch keine zugrundeliegende physische oder logische Struktur der Schlussfolgerungen. Als Konsequenz ist dieses Schema als unflexibel gegenüber modifizierten oder fehlerhaften Eingaben zu bewerten. Allens Schema richtet sich nach den Gesetzen der Logik, ignoriert jedoch nützliche Gesetzte der Zeit-Physik. Diese fehlenden Gesetzte können mit Hilfe des Ansatzes von FREKSA (1992) hinzugefügt werden. Freksa nutzt das Verfahren der konzeptionellen Nachbarschaft (conceptual neighborhood). Beziehungen zwischen Eventpaaren sind konzeptionelle Nachbarn, wenn sie direkt durch eine Verformung (z.B. Verkürzung oder Verlängerung) in eine andere Relation transformiert werden können. So sind die Relationen before (<) und meets (m) konzeptionelle Nachbarn, da eine zeitliche Verlängerung des ersten Events einen Übergang in das zweite Event ermöglicht. Die Relationen before (<) und overlaps (o) sind keine konzeptionellen Nachbarn, da dieses beispielsweise nur durch eine Verbindung (meets) erreicht werden kann. Freksas Prinzip der konzeptionellen Nachbarn ist in Abbildung 9-5 gezeigt [FREKSA, 1991].

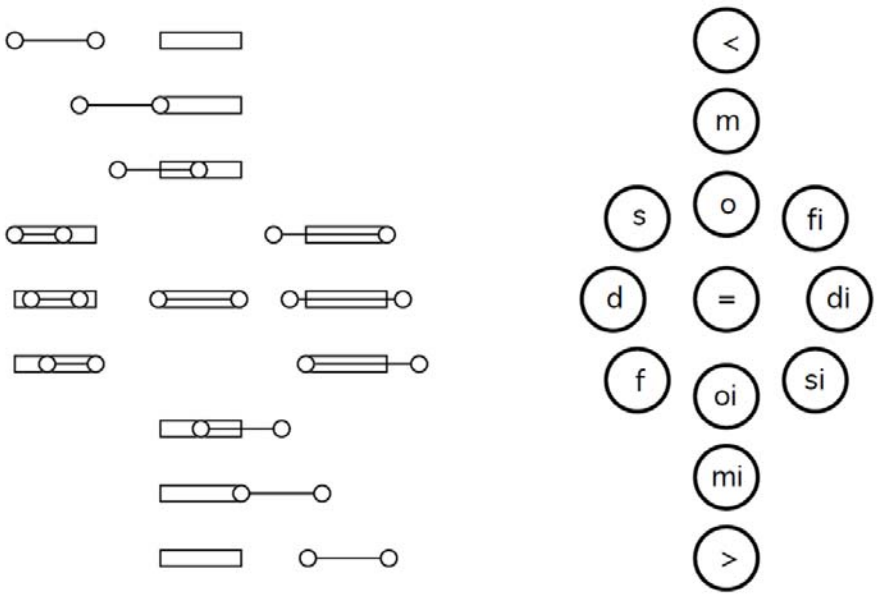


Abbildung 9-5: Konzeptionelle Nachbarschaft [FREKSA, 1992]

Mit Hilfe der konzeptionellen Nachbarn lassen sich zudem die von Allen beschriebenen Schlussfolgerungen der 13 Relationen (siehe Symbole der Freksa Notation in Abbildung 9-6) in 18 neue Relationen umwandeln und beschreiben (vgl. auch Anhang **K: Allen und Freksa** Schlussfolgerungen): older (ol), head to head with (hh), younger (yo), survived by (sb), tail to tail with (tt), survives (sv), precedes (pr), born before death of (bd), contemporary of (ct), died after birth of (db), suceeds (sd), older and survived by (ob), older contemporary of (oc), surviving contemporary of (sc), survived by contemporary of (bc), younger contemporary of (yc), younger and survives (ys) und no information (?). Mit diesen neuen Relationen (somit 31) können erneut Schlussfolgerungen gezogen werden (vgl. Anhang K: Allen und Freksa Schlussfolgerungen).

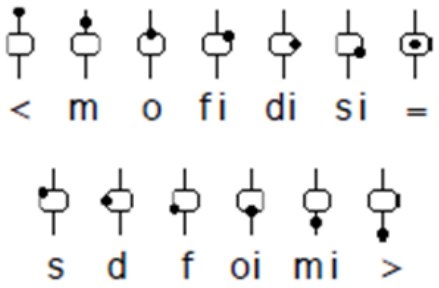


Abbildung 9-6: Allen Relationen in Freksa Symbolen [FREKSA, 1992]

Die temporale Logik nach Allen und Freksa stellen die Basis der in Kapitel 9.2 vorgestellten einfachen Modellierung einer Ontologie relativchronologischer Bezüge dar. Schlussfolgerungen und Regeln werden jedoch nicht berücksichtigt. Ein Problem der relativ chronologischen Modellierung kann die fehlende Länge eines Intervalls, bzw. die Periodendauer sein. Ist diese nicht bekannt, ist es äußerst schwierig die passende Relation auszuwählen.

Allens Intervallalgebra zeigt Ähnlichkeiten zum 9-Intersection-Model[507] (siehe Abbildung 9-7). Dieses topologische Modell zeigt räumliche Beziehungen (2 Geometrien in 2 Dimensionen) und wird vor allem in der Auswertung von Geoinformationssystemen genutzt (EGENHOFER, 1993).

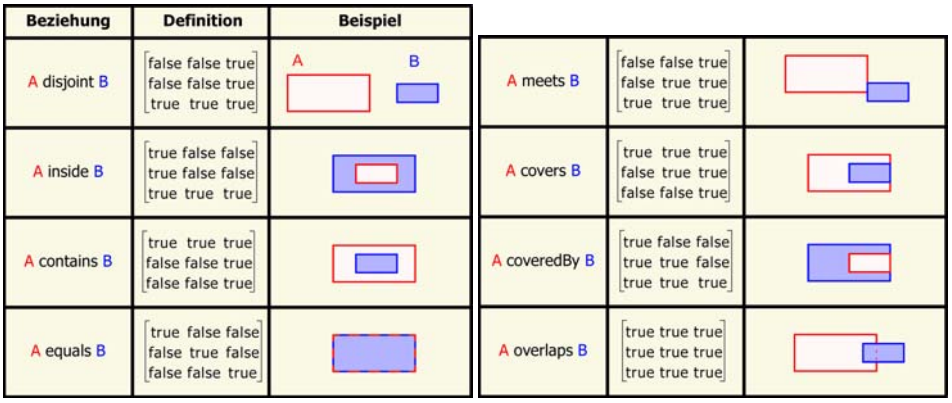


Abbildung 9-7: 9-Intersection-Model [FERGI, 2013]

Tabelle 9-2 zeigt einen Vergleich des in Abbildung Abbildung 9-7 gezeigten Modells mit den Allen Relationen:

Tabelle 9-2: Vergleich Allen Relationen / 9-Intersection-Model

9-Intersection-Model	Allen Relation
disjoint	<, >
equals	=
inside	d
contains	di
meets	m, mi
covers	f, s
coverdBy	keine Entsprechung
overlaps	o, oi
keine Entsprechung	si, fi

[507] siehe auch http://en.wikipedia.org/wiki/DE-9IM (abgerufen 04.12.2013)

9.2 Ontologie relativ chronologischer Bezüge (Prototyp)

Aus der in Kapitel 9.1 gezeigten temporalen Logik von Allen und Freksa kann eine einfache Prädikatenliste als RDF und RDFs erstellt werden. Die temporale Logik bietet alle Möglichkeiten, so dass alle erdenklichen relativchronologischen Bezüge, inkl. der Schlussfolgerungen modelliert und abgefragt werden können. Regeln zur Unterbindung nicht logischer Beziehungen sind noch nicht implementiert. Durch die Nutzung dieser chronologischen Bezüge können alle damit einhergehenden Schlussfolgerungen aus der Literatur übernommen werden. Des Weiteren wird ein Ansatz von SKOS genutzt.

```
1  # Definition von Prädikaten als rdf:type
2  #(keine range und domain, da für alles offen)
3
4  # Prefix
5  @prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
6  @prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
7  @prefix time:    <http://143.93.114.104/timeexplorer/timeVocabulary#> .
8
9  #Allen
10 time:TimeIntervals  rdf:type skos:Concept;
11  skos:prefLabel "TimeIntervals"@en;
12  skos:prefLabel "Zeitintervalle"@de;
13  skos:definition "Time Intervals Ontology according
                     to Allen and Freksa"@en;
14  skos:definition "Zeitintervallontologie nach Allen und Freksa"@de;
15 time:before                      rdf:type      rdf:Property .
16 time:after                       rdf:type      rdf:Property .
17 time:during                      rdf:type      rdf:Property .
18 time:contains                    rdf:type      rdf:Property .
19 time:overlaps                    rdf:type      rdf:Property .
20 time:overlapped-by               rdf:type      rdf:Property .
21 time:meets                       rdf:type      rdf:Property .
22 time:met-by                      rdf:type      rdf:Property .
23 time:starts                      rdf:type      rdf:Property .
24 time:started-by                  rdf:type      rdf:Property .
25 time:finishes                    rdf:type      rdf:Property .
26 time:finished-by                 rdf:type      rdf:Property .
27 time:equals                      rdf:type      rdf:Property .
```

```
28
29 #Semi-Intervals (Freska)
30 time:older                      rdf:type      rdf:Property .
31 time:headToHeadWith             rdf:type      rdf:Property .
32 time:younger                    rdf:type      rdf:Property .
33 time:survivedBy                 rdf:type      rdf:Property .
34 time:tailToTailWith             rdf:type      rdf:Property .
35 time:survives                   rdf:type      rdf:Property .
36 time:precedes                   rdf:type      rdf:Property .
37 time:bornBeforeDeathOf          rdf:type      rdf:Property .
38 time:contemporaryOf             rdf:type      rdf:Property .
39 time:diedAfterBirthOf           rdf:type      rdf:Property .
40 time:OlderAndSurvivedBy         rdf:type      rdf:Property .
41 time:OlderContemporaryOf        rdf:type      rdf:Property .
42 time:survivingContemporaryOf    rdf:type      rdf:Property .
43 time:survivedByContemporaryOf   rdf:type      rdf:Property .
44 time:youngerContemporaryOf      rdf:type      rdf:Property .
45 time:youngerAndSurvives         rdf:type      rdf:Property .
```

Um das Verständnis der Begrifflichkeiten zu verbessern, wird im Time Explorer (Kapitel 9.4) ein deutscher Namenraum verwendet. Zur Vollständigkeit sei dieser nachfolgend aufgelistet:

```
1  # Definition von Prädikaten
2
3  # Prefix
4  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
5  @prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
6  @prefix time:<http://143.93.114.104/timeexplorer/timeVocabulary#> .
7
8  #Allen
9  time:vor                        rdf:type     rdf:Property .
10 time:nach                       rdf:type     rdf:Property .
11 time:waehrend                   rdf:type     rdf:Property .
12 time:beinhaltet                 rdf:type     rdf:Property .
13 time:startetVorUndEndetIn       rdf:type     rdf:Property .
14 time:startetInUndEndetNach      rdf:type     rdf:Property .
```

```
15 time:startetVorUndEndetMit        rdf:type     rdf:Property .
16 time:startetMitUndEndetNach       rdf:type     rdf:Property .
17 time:startetMitUndEndetIn         rdf:type     rdf:Property .
18 time:startetInUndEndetNach        rdf:type     rdf:Property .
19 time:startetInUndEndetMit         rdf:type     rdf:Property .
20 time:staretVorUndEndetNach        rdf:type     rdf:Property .
21 time:gleich                       rdf:type     rdf:Property .
```

9.3 Visualisierungsstrategien und Evaluation

Die in Kapitel 9.2 vorgestellte relativchronologische RDF-Ontologie nach Allen und Freksa ergibt natürlicherweise einen RDF-Graphen. Es stellt sich jedoch die Frage, in welcher Art und Weise relativ chronologische Bezüge visualisiert werden können, so dass es den Geisteswissenschaften hilft, auch komplexere Chronologien und Abhängigkeiten zu verstehen und bei Modifizierungen der Chronologie Änderungen schnell zu erkennen. Das Finden der richtigen Visualisierungsstrategien ist ein komplexer Vorgang, das an einem kleinen Beispiel verdeutlicht werden kann. ORTIZ (2012) zeigt an den Zahlen 75 und 37, 45 verschiedene Möglichkeiten der Visualisierung auf. Nachfolgend (vgl. Abbildung 9-8) sind einige Beispiele abgebildet.

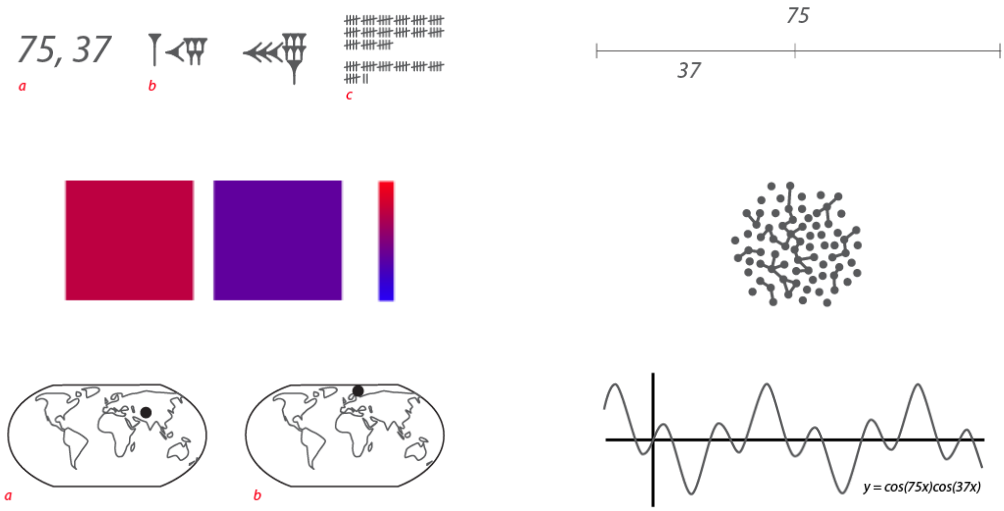


Abbildung 9-8: Darstellung der Zahlen 37 und 75 [ORTIZ, 2012]

Abbildung 9-8 verdeutlicht die Schwierigkeit des Findens der richtigen Visualisierungsstrategie. Die Zahlen 75 und 37 könnten sowohl als arabische Ziffern, babylonische Zeichen[508], bzw. Striche (link oben), als Proportionen (rechts oben), in einer Farbskala (mitte links), mit Hilfe eines Netz-

[508] siehe auch http://www-history.mcs.st-and.ac.uk/HistTopics/Babylonian_numerals.html (04.12.2013)

werks / Graphen (mitte rechts), durch geographische Koordinatenpaare (unten links) oder als mathematische Funktion (unten rechts) beschrieben werden. Je nach Anwendungsfall kann die ein oder andere Visualisierungsstrategie das passende Ergebnis erzielen. Internetplattformen wie www.datavisualization.ch geben einen Überblick zu verschiedenen Tools, die eine Visualisierung vereinfachen können.

Chronologien werden üblicherweise mit Hilfe von Zeitleisten bzw. Zeitstrahlen beschrieben, siehe zum Beispiel Kindred Britain[509] (vgl. Abbildung Abbildung 9-9) oder auch eine Darstellung der christlich jüdischen Geschichte[510]. Diesen Zeitleisten liegen absolute Datumswerte zu Grunde. Sie sind im Allgemeinen zur Repräsentation von zeitlichen Ereignissen anerkannt, in den Geisteswissenshaften weit verbreitet und leicht verständlich. Insbesondere Gantt-Diagramme[511] können z.B. einen Projektverlauf verdeutlichen.

Abbildung 9-9: Kindred Britain [KINDREDBRITAIN, 2013]

RDF-Graphen und damit die in Kapitel 9.2 beschriebene Ontologie, verfolgen ein anderes Konzept. Hierbei werden lediglich Beziehungen (Knoten und Kanten) zwischen verschiedenen Objekten modelliert, so dass ein gerichteter Graph entsteht. Absolute Datenwerte sind in diesem Fall nicht

[509] vgl. http://kindred.stanford.edu (abgerufen 04.12.2013)

[510] vgl. http://www.simile-widgets.org/timeline/examples/religions/religions.html; Visualisierungen mit Hilfe von http://www.simile-widgets.org/timeline/ (abgerufen 04.12.2013)

[511] siehe auch http://ganttdiagramm.com (abgerufen 04.12.2013)

nötig. Frameworks wie D3 (Data-Driven Documents)[512] ermöglichen eine Vielzahl an Repräsentation von Graphen[513] (vgl. auch Abbildung 9-10).

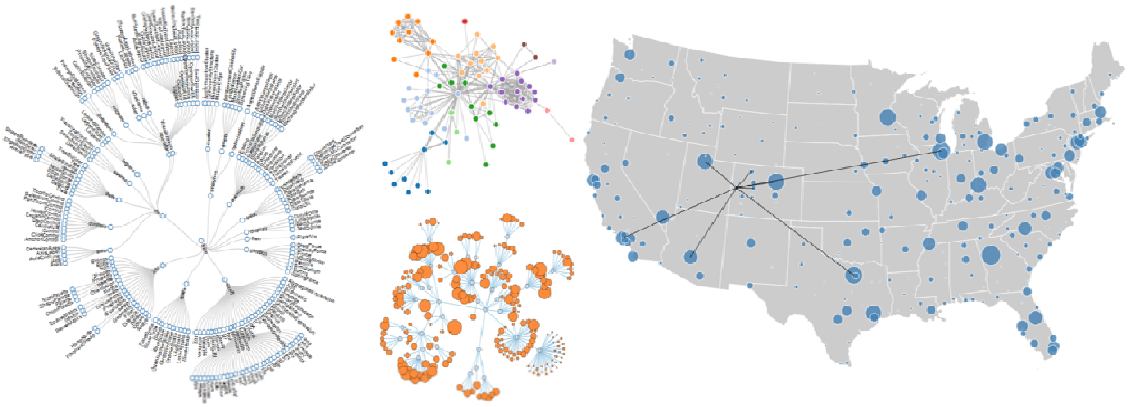


Abbildung 9-10: Graphen

Insbesondere RDF-Graphen können mit Hilfe von Frameworks dargestellt werden. Abbildung 9-11 zeigt den Graphen des Pleiades Places Langenhain[514] mit Hilfe der Bibliothek VisualRDF[515]. Eine chronologische Abbildung durch Graphen ist jedoch für den Menschen untypisch und kann nur schwer verstanden werden.

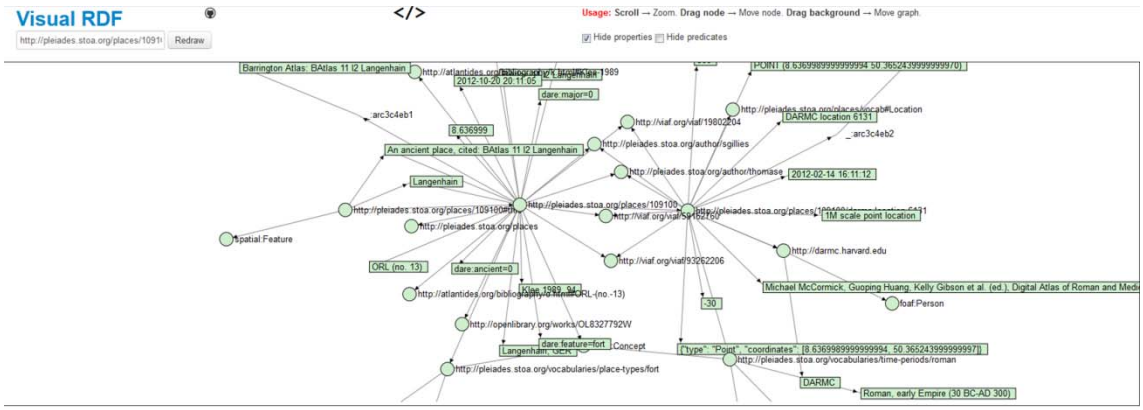


Abbildung 9-11: Pleiades Place Langenhain mit VisualRDF[516]

Eine dritte Möglichkeit der Darstellung bietet der Text. Chronologische Beziehungen können über einfache, verständliche 'Sätze' oder auch Tripel (ein Satz aus Subjekt, Prädikat und Objekt[517]) be-

---

[512] vgl. https://github.com/mbostock/d3 (abgerufen 04.12.2013)

[513] siehe https://github.com/mbostock/d3/wiki/Gallery (abgerufen 04.12.2013)

[514] vgl. http://pleiades.stoa.org/places/109100 (abgerufen 04.12.2013)

[515] vgl. https://github.com/alangrafu/visualRDF (abgerufen 04.12.2013)

[516] Quelle: http://graves.cl/visualRDF/?url=http://pleiades.stoa.org/places/109100/turtle (04.12.2013)

reitgestellt werden, so dass sich dem Nutzer der Sinn aus dem Kontext erschließt. Liegt jedoch eine große Datenmenge vor, kann dies leicht unverständlich werden und die Übersicht verloren gehen.

Darüber hinaus kann und sollte über weitere Visualisierungsformen nachgedacht werden. Tabelle 9-3 stellt die drei favorisierten Visualisierungsstrategien relativ chronologischer Bezüge mit Vor- und Nachteilen gegenüber:

Tabelle 9-3: Visualisierungsstrategien für relativ chronologische Bezüge

Visualisierungsart	**Vorteile**	**Nachteile**
Zeitleiste	allgemein anerkannt, leicht lesbar, Frameworks vorhanden	absolute Zahlen notwendig, (Übersichtlichkeit bei großen Datenmengen geht evtl. verloren)
Graph	keine absolute Zahlen notwendig, direkt aus RDF ableitbar, Frameworks vorhanden	geben nur Beziehungen wieder, schwer lesbar, nicht für Zeitabfolgen anerkannt, (Übersichtlichkeit bei großen Datenmengen geht evtl. verloren)
Text	keine absoluten Zahlen nötig, direkt aus RDF ableitbar, keine Frameworks nötig	geben nur Beziehungen wieder, schwer lesbar, Übersichtlichkeit bei großen Datenmengen geht schnell verloren

Betrachtet man die 13 Relationen von Allen, stellt sich die Frage, wie diese Relationen in einer Zeitleiste oder einem Graphen abgebildet werden können, so dass deren Logik auf die Darstellungen übertragen werden und dem Nutzer das Lesen der Abbildung leicht fällt (weitere Visualisierungsideen sind Anhang L: Visualisierungsideen zu entnehmen). In Abbildung 9-12 sind alle Allen-Relationen in Form einer Zeitleiste dargestellt. Folgen zwei Events direkt nacheinander (m, mi, <, >), ist eine Darstellung in einer Zeile möglich. In allen anderen Fällen muss auf eine andere Zeile ausgewichen werden. Zur besseren Unterscheidung der Relationen können zudem Farben und verschiedene Stricharten eingesetzt werden, die verschiedene Aspekte eines Events zu einem Referenzevent (z.B. Event hat den gleich Start- bzw. Endpunkt und eine Überlappung mit dem Referenzevent →Farbe: grün) darstellen. Durch diese Unterscheidung wird ein leichtes Verständnis erzeugt. Ein Problem dieser Darstellung ist die Erzeugung absoluter Zeitangaben, die für eine Visualisierung dringend nötig sind. Hierbei muss eine definierte Länge einer Zeitperiode (die Länge ist nicht bekannt) vereinbart und ein Referenzdatum gesucht werden, auf das alle Bezüge referenziert werden müssen. Die in der Abbildung aufgezeigten räumlichen Lagen der Zeitperioden sind ebenfalls für eine gut lesbare Darstellung wichtig (z.B. gleicher Startpunkt, endet in der Mitte der Referenz, etc.).

[517] siehe auch http://de.wikipedia.org/wiki/Subjekt-Verb-Objekt (abgerufen 04.12.2013)

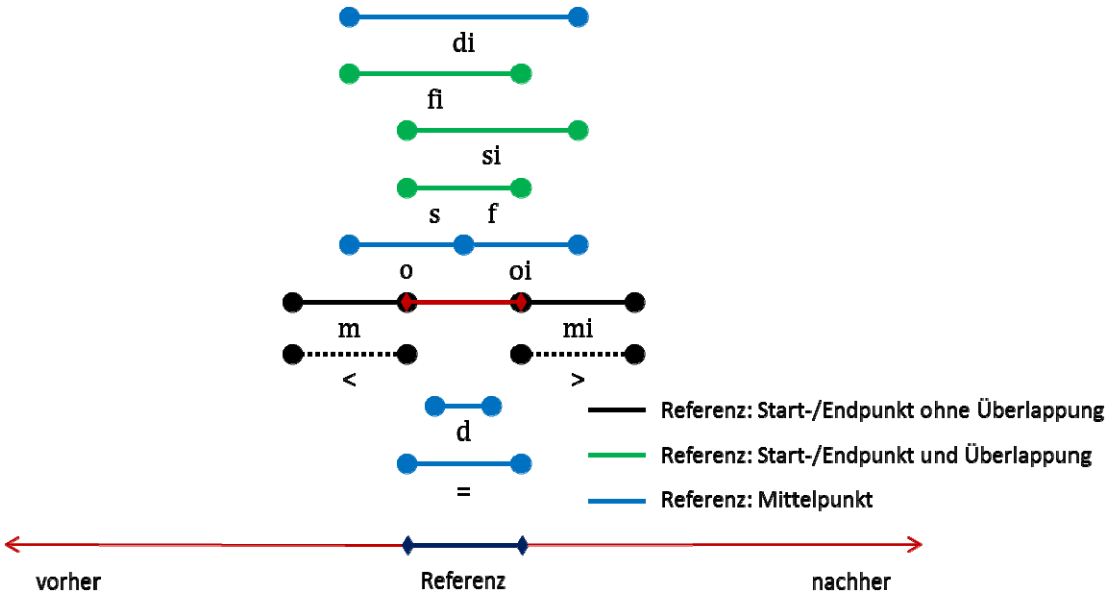


Abbildung 9-12: Visualisierungsidee Allen Relationen (Zeitleiste)

Die Darstellung der Relationen in einem Graphen gestaltet sich als schwierig. Es ist keine Zeitleiste (mit absoluten Daten) vorhanden, die eine Richtung vorgeben kann. Zur besseren Verständlichkeit sollte die Richtung der Graphen-Elemente (EventA ist nach Event B, so ist EventA links von EventB darzustellen) einer Zeitleiste nachempfunden werden. Eine verständliche und einfache Lösung der Visualisierung von Kanten und Knoten der Relationen after und before ist möglich. Die Relationen meets und met-by können ebenfalls links und rechts des Referenzevents dargestellt werden. Damit Verwechslungen mit den Relationen after und before ausgeschlossen werden können, werden diese länger und gepunktet dargestellt. Alle anderen Darstellungen von Relationen sind sehr schwierig zu realisieren. Die Richtungen der übrigen Relationen sind Abbildung 9-4 nachempfunden. Eine Unterscheidung wird in diesem Fall durch verschiedene Winkel und Farben, sowie Stricharten vorgenommen. Inversen der Relationen sind auch im Graphen als Inversen dargestellt, so dass die innere Logik des Graphen nicht geändert wird. Auch hier muss eine Referenzlänge der Kanten bestimmt werden, die Fixierung eines Referenzzeitpunktes als absolute Zahl entfällt jedoch.

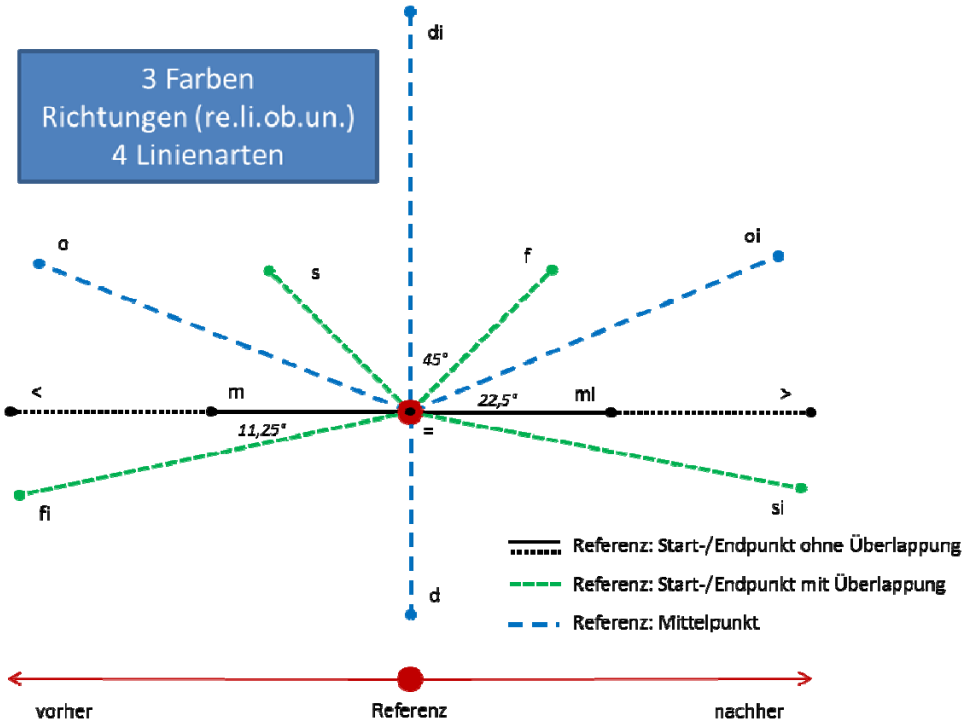


Abbildung 9-13: Visualisierungsidee Allen Relationen (Graph)

Die abstrakten Visualisierungsstrategien eines Graphen, bzw. einer Zeitleiste werden nachfolgend mit einem Beispiel verdeutlicht. Hierzu sei ein Satz an Tripeln gegeben, die in einen RDF-Graphen (vgl. Abbildung 9-14) münden:

```
A mb Ref. B m Ref. D mb Ref. F d B. I < B.
J f B. G d F. K di J. H mb G. L mb K.
M si L. E s D. N > M. O o N. C o O. P oi O.
```

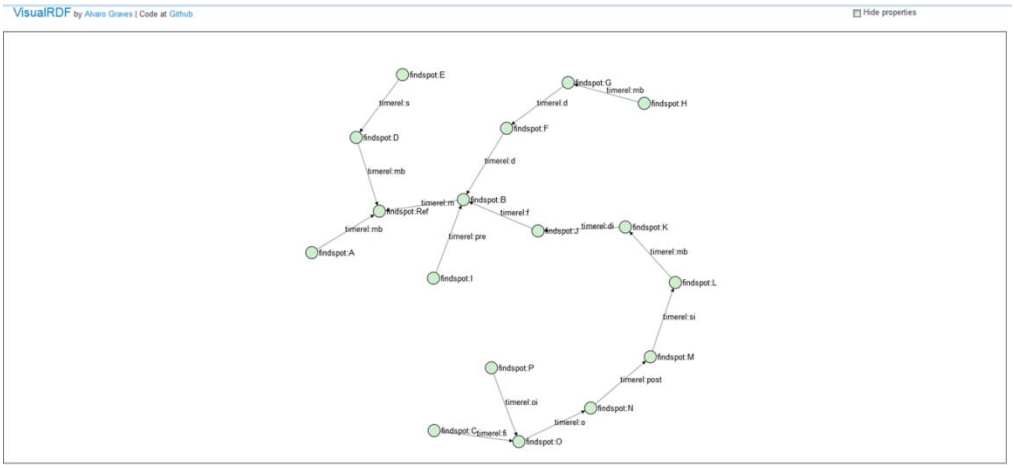


Abbildung 9-14: Allen Relationen Beispiel (RDF-Graph)

Abbildung 9-15 zeigt ein einfarbiges Beispiel (links) des Satzes an Tripels als Timeline. Vergleicht man diese Ansicht mit dem zuvor gezeigten RDF-Graphen, wird deutlich, dass die Beziehungen nicht eindeutig ersichtlich sind. Dies könnte durch eine Einfärbung der verschiedenen 'Beziehungsknoten' gelöst werden (rechts). Negativ ist jedoch zu bewerten, dass beispielsweise das Event A fast gleichzeitig mit Event O dargestellt wird. Dies stellt eine Schwäche der Visualisierung dar, da diese Informationen verwirren und nicht der Wahrheit entsprechen kann. Eine weitere Einfärbung, wie zuvor vorgeschlagen (z.B: Event hat den gleich Start- bzw. Endpunkt und eine Überlappung mit dem Refenzevent →Farbe: grün) ist hier ebenfalls nicht hilfreich. An diesem kleinen übersichtlichen Beispiel lässt sich zeigen, dass die Darstellung von relativ chronologischen Beziehungen auf einer Timeline sehr schwierig und nicht intuitiv ist. Das Verständnis der Komplexität des Sachverhalts muss mit weiteren helfenden Mitteln erfolgen.

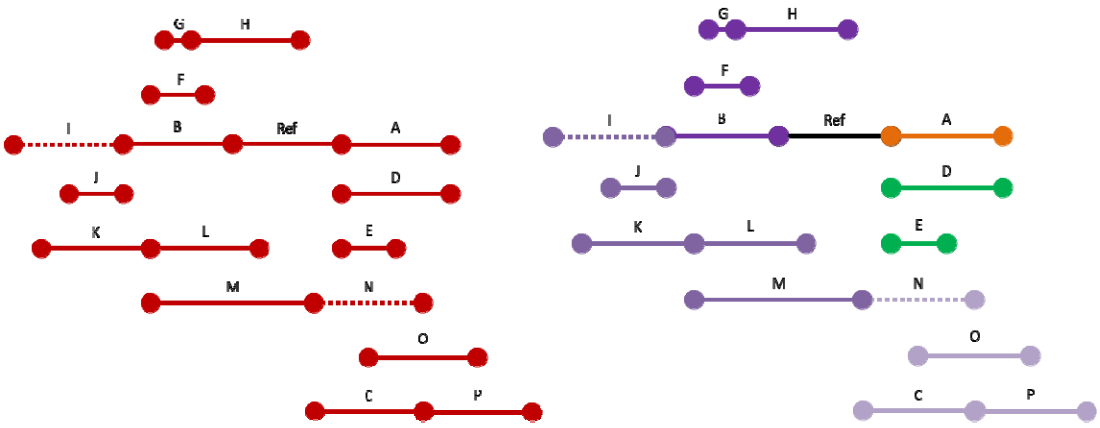


Abbildung 9-15: Allen Relationen Beispiel (Zeitleisten)

Die gleiche Situation wird nun mit einem Graphen, nach den zuvor gezeigten Gestaltungsregeln abgebildet (vgl. Abbildung 9-16). Hier ist zu erkennen, dass zwei Probleme auftreten: Zum einen muss bei der Verwendung von mehrfachen Relationen (hier zweimal met-by) z.B. der Winkel verändert werden, so dass beide Relationen nicht übereinander liegen, zum anderen kann durch besondere Umstände ebenfalls eine visuelle Dopplung von Kanten erfolgen. Diese Ansicht ist sehr ungewohnt und könnte den Geisteswissenschaftler sehr verwirren. In diesem Fall werden mit dem Graphen die Beziehungen der Events untereinander optimal dargestellt, die zeitliche Abfolge ist nur schwer zu erkennen.

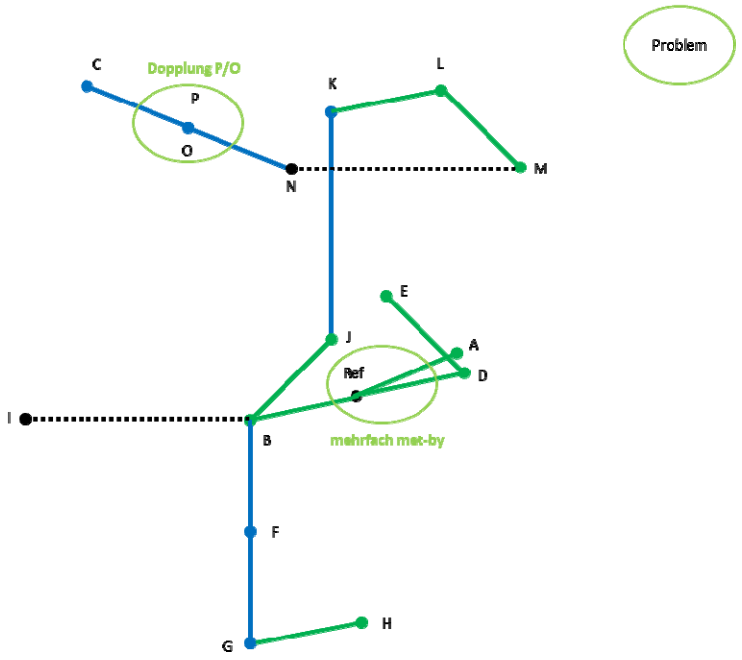


Abbildung 9-16: Allen Relationen Beispiel (Graph)

Betrachtet man dieses kleine Beispiel, sind zwei wichtige Erkenntnisse möglich: Eine Darstellung von relativchronologischen Beziehungen mittels eines Graphen oder einer Zeitleiste ist sehr schwierig umzusetzen. Graphen, wie auch Zeitleisten, bieten Vorteile, jedoch auch erhebliche Nachteile. Insbesondere Graphen können die Beziehungen der einzelnen Events untereinander gut darstellen, deren zeitliche Einordnung ist jedoch nur über eine Zeitleiste gut zu überblicken. Die textliche Repräsentation alleine kann ebenfalls keinen hinreichenden Überblick bieten. Aufgrund dessen wird in der prototypischen Umsetzung der Visualisierung (vgl. Kapitel 9.4) auf eine Lösung mit Hilfe aller drei Visualisierungsformen gesetzt. So können die Nachteile des einen durch Vorteile des anderen ausgeglichen werden.

9.4 Der Time Explorer

Mit dem Time Explorer wird ein Prototyp der vorgestellten Visualisierungsstrategien (vgl. Kapitel 9.3) und der relativchronologischen Ontologie (vgl. Kapitel 9.2) umgesetzt. In der PostGIS Datenbank sind absolute Datumsangaben in der Tabelle der Findspots abgelegt. Diese absoluten Angaben werden jedoch nicht direkt durch die Findspots bestimmt. Diese Daten sind aggregiert, so dass direkte chronologische Schlüsse nicht gezogen werden können. Aufgrund dieses Umstands wird in diesem Prototypen ein kleines überschaubares Beispiel genutzt. Auch an diesem Beispiel können Simulationen (Veränderungen der Beziehungen) getätigt und deren Auswirkungen betrachtet werden. In diesem Beispiel sind absolute Zeitangaben von zufällig ausgesuchten Findspots (inkl. einer Referenz, Pompeii[518]) in relative Beziehungen übertragen worden, siehe auch Abbildung 9-17[519].

[518] siehe auch http://de.wikipedia.org/wiki/Pompeji (abgerufen 04.12.2013)

- Friedberg (10-20)
- Camulodunum (10-65)
- La_Graufesenque__Fosse_Cirratus (30-40)
- Rheingoenheim (40-73)
- Pompeii (unbekannt-79)
- Castleford (71-105)
- Riemst__Tumulus (135-150)

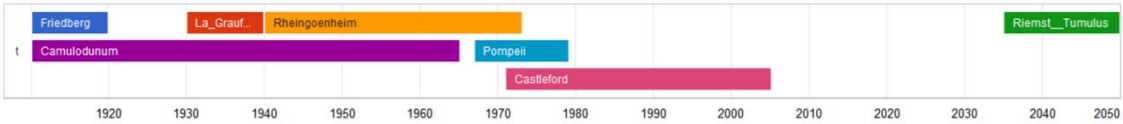


Abbildung 9-17: Time Explorer Beispiel (absolute Zeiten)

Aus diesen Informationen lassen sich unter Anderem nachfolgende Tripels, nach Vorgabe, der in Kapitel 9.2 beschriebenen Ontologie, bilden. Es ist zu erkennen, dass z.B. La Grausesenque, Fosse Cirratus als meets Rheingönheim modelliert ist. Dies könnte jedoch auch auch als during Camulodunum modelliert werden. Somit zeigt sich, dass eine Modellierung der relativchronologischen Bezüge anhand weiterer quantitativer Informationen (z.B. räumliche Nähe, gleiches Vorkommen an Gefäßformen, etc.) erfolgen muss. In diesem Beispiel wird dies jedoch nicht weiter verfolgt.

```
1 @prefix time: <http://143.93.114.104/timeexplorer/timeVocabulary#> .
2 @prefix site: <http://143.93.114.104/rest/samian/findspots/> .
3 site:Rheingoenheim time:overlaps site:Pompeii .
4 site:Castleford time:overlapped-by site:Pompeii .
5 site:Riemst__Tumulus time:after site:Castleford .
6 site:Camulodunum time:overlaps site:Rheingoenheim .
7 site:Friedberg time:starts site:Camulodunum .
8 site:La_Graufesenque__Fosse_Cirratus time:meets site:Rheingoenheim .
```

Die Modellierung der Chronologie folgt den nachstehenden Prinzipien. Durch die Einschränkung, dass ein Subjekt nur eine Beziehung zu einem Objekt eingehen kann werden eindeutige Relationen vorausgesetzt, z.B. nicht A vor B und A vor C. Hierbei ist die Beziehung zwischen B und C nicht klar definiert. Beziehungen wie A vor B und C nach B generieren eindeutige Verhältnisse.

[519] Hier ist anzumerken, dass für eine Darstellung mit Hilfe der Google API allen Daten der Wert 1900 aufaddiert wurde. Des Weiteren wird Pompeii aus Darstellungsgründen die Länge 3 zugewiesen.

- ein Subjekt beschreibt nur eine Beziehung zu einem Objekt (sonst wird es nicht deutbar, evtl. nicht mehr in einem Graphen)
- ein Objekt muss zuvor als Subjekt eingeführt worden sein
- ein Event kann mehrmals als Objekt auftreten
- das erste Objekt ist ein Referenzobjekt (z.B. Pompeij)
- alle Beziehungen müssen in einem Graphen abbildbar sein

Die vorliegenden Tripel sind in zwei Datenbanktabellen, triples und triples_org (id[int]PK;triple[char.variy]), in der bereits erstellten PostGIS Datenbank abgelegt (vgl. nachfolgende SQL-Statements und input_triple_samian.sql). Hierbei wird nicht der verfügbare Triplestore genutzt, da Probleme bei der Löschung eines bestimmten Tripels auftreten[520]. In der Tabelle triples_org werden die Originaltripel gespeichert. Die Tabelle triples dient zur Speicherung im aktiven Prozess, so dass Tripel bei Änderungen in der relativen Chronologie abgelegt werden können. Durch die Teilung in zwei verschiedene Tabellen ist ein Zurücksetzen der Beziehungen auf den Ursprungszustand möglich.

```
INSERT INTO triples VALUES (1,'site:Rheingoenheim time:overlaps site:Pompeii .');
INSERT INTO triples VALUES (2,'site:Castleford time:overlapped-by site:Pompeii .');
INSERT INTO triples VALUES (3,'site:Riemst__Tumulus time:after site:Castleford .');
INSERT INTO triples VALUES (4,'site:Camulodunum time:overlaps site:Rheingoenheim .');
INSERT INTO triples VALUES (5,'site:Friedberg time:starts site:Camulodunum .');
INSERT INTO triples VALUES (6,'site:La_Graufesenque__Fosse_Cirratus time:meets
site:Rheingoenheim .');
```

```
INSERT INTO triples_orig VALUES (1,'site:Rheingoenheim time:overlaps site:Pompeii .');
INSERT INTO triples_orig VALUES (2,'site:Castleford time:overlapped-by site:Pompeii .');
INSERT INTO triples_orig VALUES (3,'site:Riemst__Tumulus time:after site:Castleford .');
INSERT INTO triples_orig VALUES (4,'site:Camulodunum time:overlaps site:Rheingoenheim
.');
INSERT INTO triples_orig VALUES (5,'site:Friedberg time:starts site:Camulodunum .');
INSERT INTO triples_orig VALUES (6,'site:La_Graufesenque__Fosse_Cirratus time:meets si-
te:Rheingoenheim .');
```

Der Time Explorer ist eine Servlet-Anwendung die als timeexplorer.war im Apache-Tomcat zur Verfügung steht und durch die URL http://143.93.114.104/timeexplorer aufrufbar ist. Die Anwendung bindet einen Treiber für PostgreSQL Datenbanken über Maven ein, der den lesenden und schreibenden Zugriff auf die verwendete PostGIS Datenbank ermöglicht. Der Prototyp besteht aus

[520] Für zukünftige Anwendungen sollte dieses Problem jedoch behoben und ein Triplestore genutzt werden

zwei Klassen (PostGIS und Triple) und mehreren Servlets. Die Klasse PostGIS baut Verbindungen zur Datenbank auf, fragt den Bestand an Tripeln ab oder löscht und erstellt neue Tripels im Falle einer Änderung durch eine Simulation. In der Klasse Triple hingegen, werden die Koordinaten (Start- und Endpunkt eines jeden Events; z.B. [1900;1920]) zur Visualisierung auf einem Zeitstrahl berechnet. Die Ausführungen in Kapitel 9.3 zeigen, dass für eine Darstellung mit Hilfe einer Timeline absolut chronologische Daten vorliegen müssen.

Durch einen Algorithmus werden aus den bestehenden relativen Beziehungen absolute Datumsangaben (nachfolgend Koordinaten genannt, um zu zeigen, dass die resultierenden Werte keine echten Datumsangaben darstellen) erzeugt. Es wird ebenfalls festgelegt, dass eine durchschnittliche Periode 20 (Jahre) lang ist (dies kann z.B. durch die Relation starts in 10 Jahre verkürzt werden, vgl. Abbildung 9-4) und die Referenzperiode den Zeitraum 1900 bis 1920 einnimmt. Der Algorithmus sortiert zunächst alle Tripels so, dass ein Event erst nach der Einführung als Subjekt ein Objekt eines Tripels werden kann. Im nachfolgenden Beispiel tauschen z.B. die Triple mit Subjekt Riemst__Tummulus und Castleford die Reihenfolge.

```
site:Rheingoenheim time:overlaps site:Pompeii .
site:Camulodunum time:overlaps site:Rheingoenheim .
site:Friedberg time:starts site:Camulodunum .
site:La_Graufesenque__Fosse_Cirratus time:meets site:Rheingoenheim .
site:Riemst__Tumulus time:after site:Castleford .
site:Castleford time:overlapped-by site:Pompeii .
```

Nach dem Tausch der Reihenfolge:

```
site:Rheingoenheim time:overlaps site:Pompeii .
site:Camulodunum time:overlaps site:Rheingoenheim .
site:Friedberg time:starts site:Camulodunum .
site:La_Graufesenque__Fosse_Cirratus time:meets site:Rheingoenheim .
site:Castleford time:overlapped-by site:Pompeii .
site:Riemst__Tumulus time:after site:Castleford .
```

Dies ist insbesondere wichtig, da der Algorithmus zur Berechnung der Koordinaten die direkte Beziehung des Subjekts zu einem Objekt nutzt. Wird beispielsweise Friedberg als Subjekt und Camulodunum als Objekt eingeführt und die Relation starts verwendet, so wird vorausgesetzt, dass Camulodunum bereits Koordinaten besitzt (hier z.B. [1880;1900]) und sich somit die Friedberg-Koordinaten [1880;1890] ergeben. Das Fehlen der Camulodunum-Koordinaten führt zu einem

Abbruch[521]. Tabelle 9-4 stellt die Berechnungen dar, in Tabelle 9-5 ist ein Beispiel mit B [1900;1920] gezeigt, hierbei gilt:

- x = durchschnittliche Periodendauer = 20
- l = aktuelle Länge des Objekts
- m = aktueller Mittelwert des Objekts (l/2)
- oa = Anfangskoordinate des Objekts
- oe = Endkoordinate des Objekts
- sa = Startkoordinate des Subjekts
- se = Endkoordinate des Subjekts:

Tabelle 9-4: Berechnung der Koordinaten

Relation	**resultierendes Koordinatenpaar**
A before B [oa;oe]	A [se-x;oa-(x/2)] B [oa;oe]
A after B [oa;oe]	A [oe+(x/2);sa+x;] B [oa;oe]
A during B [oa;oe]	A [m-0.25*l;m+0.25*l] B [oa;oe]
A contains B [oa;oe]	A [m-(l/2)-0.25*l;m+(l/2)+0.25*l] B [oa;oe]
A overlaps B [oa;oe]	A [se-l;(oa+oe)/2] B [oa;oe]
A overlapped-by B [oa;oe]	A [(oa+oe)/2;sa+l] B [oa;oe]
A meets B [oa;oe]	A [se-x;oa] B [oa;oe]
A met-by B [oa;oe]	A [oe;sa+x] B [oa;oe]
A starts B [oa;oe]	A [oa;sa+(l/2)] B [oa;oe]
A started-by B [oa;oe]	A [oa;sa+1.5x] B [oa;oe]
A finishes B [oa;oe]	A [se-(l/2);oe] B [oa;oe]
A finished-by B [oa;oe]	A [se-1.5l;oe] B [oa;oe]
A equals B [oa;oe]	A [oa;sa] B [oa;oe]

[521] Dieser Algorithmus sollte in den folgenden Arbeiten verbessert und erweitert werden.

Tabelle 9-5: Berechnung der Koordinaten (Beispiel)

Relation	resultierendes Koordinatenpaar
A before B [1900;1920]	A [1860;1890] B [1900;1920]
A after B [1900;1920]	A [1930;1950] B [1900;1920]
A during B [1900;1920]	A [1905;1915] B [1900;1920]
A contains B [1900;1920]	A [1895;1925] [1900;1920]
A overlaps B [oa;oe]	A [1890;1910] [1900;1920]
A overlapped-by B [oa;oe]	A [1910;1930] [1900;1920]
A meets B [oa;oe]	A [1880;1900] [1900;1920]
A met-by B [oa;oe]	A [1920;1940] [1900;1920]
A starts B [oa;oe]	A [1900;1910] [1900;1920]
A started-by B [oa;oe]	A [1900;1930] [1900;1920]
A finishes B [oa;oe]	A [1910;1920] [1900;1920]
A finished-by B [oa;oe]	A [1890;1920] [1900;1920]
A equals B [oa;oe]	A [1900;1920] [1900;1920]

Nach erfolgter Berechnung der Koordinaten können Zeitleisten erzeugt werden. Hierzu dienen zwei Frameworks Google Charts Timeline[522] und Simile Widgets Timeline[523], die eine einfache Erstellung von Zeitleisten ermöglichen. Mit Simile Widgets können zudem weitere Einstellungen[524] getätigt werden (z.B. Änderung jeder Farbe einer Zeitperiode), sodass der Time Explorer Simile Widgets Timeline zur Darstellung nutzt. Der Abruf der der Timeline wird über das Servlet getTimeline (Google) bzw. getSimile (Simile Widget) realisiert (vgl. Abbildung 9-18).

[522] siehe https://developers.google.com/chart/interactive/docs/gallery/timeline (abgerufen 04.12.2013)

[523] siehe http://www.simile-widgets.org/timeline (abgerufen 04.12.2013)

[524] siehe auch http://simile-widgets.org/wiki/Timeline (abgerufen 04.12.2013)

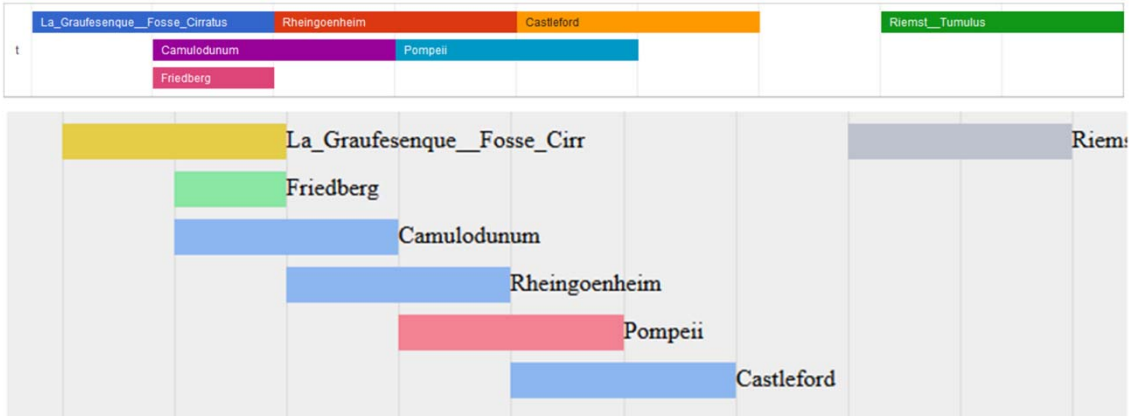


Abbildung 9-18: Google vs. Simile Timeline

Trotz gleicher Datengrundlage, die mit dem Servlet (/getTimes) abgerufen werden kann, entstehen unterschiedliche Visualisierungen. In der Simile Widgets (unten) Darstellung werden die einzelnen Events unterschiedlich eingefärbt (rot = Referenz, grau = after; before, grün = starts; finishes; started-by; finished-by; gelb = meet; met-by; blau = during; contains; overlaps; overlapped-by). Wie bereits in Kapitel 9.3 beschrieben, kann eine thematisch, farbige Darstellung helfen, die Beziehung zu einem anderen Event zu beschreiben. Es zeigen sich jedoch auch Probleme. Vergleicht man das Resultat in Abbildung 9-18 mit den ursprünglichen Daten (vgl. Abbildung 9-17), ist zu erkennen, dass das Event La Graufesenque, Fosse Cirratus während Camulodunum stattfindet. In der relativen Abbildung jedoch ist dies nicht der Fall. Hier können falsche Schlussfolgerungen gezogen werden. Somit zeigt sich, dass nicht vorhandene Periodenlängen das Ergebnis verfälschen können und weitere Visualisierungen benötigt werden, so dass ein besserer Überblick geschaffen wird. Zur Weiterverarbeitung steht ebenfalls das resultierende RDF Dokument in Turtle Systax (/getTurtle) oder RDF/XML (/getRDF) zur Verfügung. Das implementierte Zeitenvokabular kann durch das Servlet timeVocabulary abgerufen werden.

Als zweite Visualisierungsform wird ein RDF-Graph genutzt. Hierzu wird die Bibliothek VisualRDF[525] verwendet. Visual RDFbaut auf diversen Bibliotheken auf (darunter auch D3) und nutzt die Skriptsprache PHP. Eine Kombination von PHP und JAVA ist auf einem Apache Tomcat mit Komplikationen verbunden[526]. Das frei verfügbare Tool VisualRDF wird somit auf einem normalen Webserver (http://rdf.florian-thiery.de) gehostet und kann über einen Iframe in den Time Explorer eingebunden werden. Mit dem Aufruf des Servelts getTurtle wird der aktuelle RDF Graph in Turtle Notation zurückgegeben und gleichzeitig eine temporäre Datei auf dem GeInArFa Server erzeugt (/usr/share/apache-tomcat-7.0.41/webapps/share/tmp.ttl). Diese Binärdatei kann durch VisualRDF einfach ausgelesen und visualisiert werden. Aufgrund der Sicherheitsbeschränkungen des GeInArFA Servers muss jedoch eine Anmeldung mit Autorisierung erfolgen. Der aktuelle Graph kann mit dem Aufruf:http://rdf.florian-

[525] https://github.com/alangrafu/visualRDF (abgerufen 04.12.2013)

[526] Diese Komplikationen sind lösbar, jedoch aufgrund des erhöhten Aufwands nicht in dieser Arbeit nicht gelöst.

thiery.de/?url=http://benutzername:passwort@143.93.114.104/ share/tmp.ttl erzeugt werden, vgl. Abbildung 9-19.

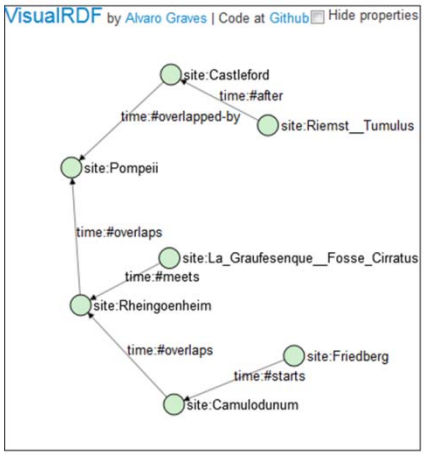


Abbildung 9-19: RDF-Graph des Beispiels

Die zwei Komponenten, Zeitleiste und der RDF-Graph, münden in der Anwendung des Time Explorers (vgl. Abbildung 9-20). In diesem ist zudem die Textansicht implementiert (/getStatus). Alle drei Komponenten sind nebeneinander (mit Hilfe von Iframes) dargestellt und ermöglichen jederzeit eine Listen-, Graph- oder Zeitleistenansicht.

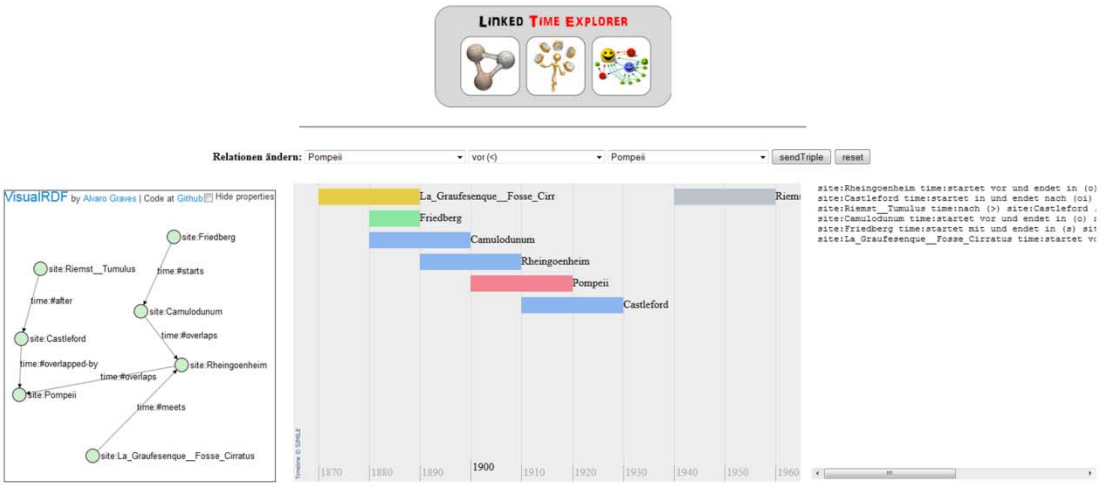


Abbildung 9-20: Linked Time Explorer

Simulationen, das heißt eine Modifizierung einer chronologischen Relation, sind über ein einfaches Interface möglich. Durch JavaScript und jQuery werden die in dem Graphen vorhanden Events (/getEntities) und möglichen Relationen (/getRealtions) geladen. Die Relationen und der Text sind in die deutsche Sprache übersetzt. Wählt man EventA, eine Relation und EventB aus, sendet das Triple an das Servelt (/sendTriple?triple=prefix:EventA Relation EventB) wird das Triple in der Datenbank geändert und eine neue Ansicht mit den Modifizierungen geladen, z.B.

Rheingönheim meets Pompeii, anstatt Rheingönheim overlaps Pompeii (vgl. Abbildung Abbildung 9-21). Man kann erkennen, dass die alle Beziehungen, die mit Rheingönheim verknüpft sind, mit Rheingönheim verschoben werden. Der Linked Time Explorer ist jedoch noch nicht gegen unkorrekte Eingaben (d.h. Relationen, die im Graphen zu Widersprüchen führen) abgesichert. Dies sollte in kommenden Versionen ebenfalls abgefangen werden.

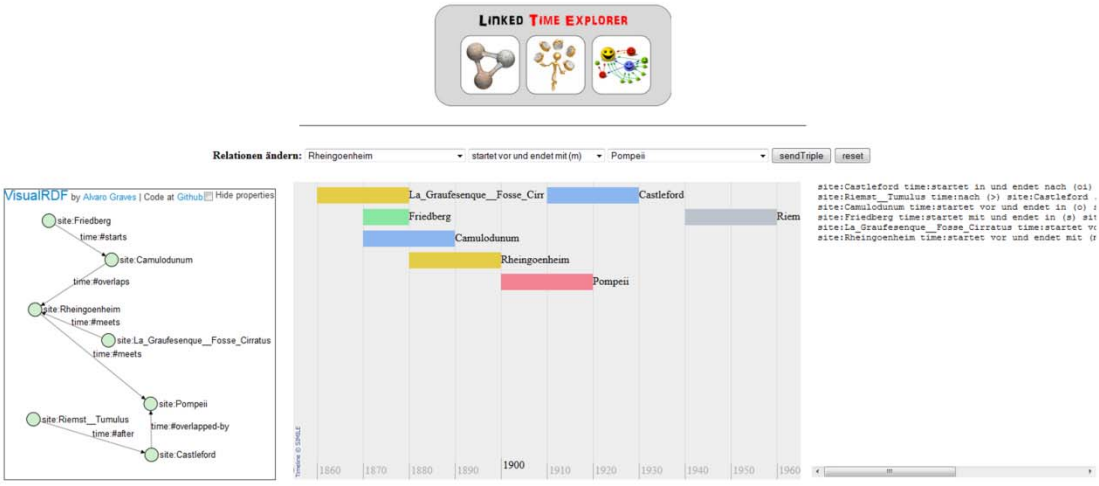


Abbildung 9-21: Linked Time Explorer nach Simulation

9.5 Zeitmodellierung in der Samian Ware

In den zuvor gezeigten Kapiteln werden auf Grund der Komplexität in der Zeitmodellierung der Samian Ware einfache Beispiele verwendet. Das absolut datierte Datum, der in der Datenbank vorliegenden Findspots, wird anhand verschiedener Faktoren ermittelt. Darunter gehören u. A. ein real datierter Site und eine Datierung mit Hilfe der Gefäßformen und verschiedenen Töpfer. Dies bedeutet, dass die Datierungen durch Aggregationen erfolgen, welche nicht in der Datenbank abgebildet und dadurch nur schwer Schlussfolgerungen zu ziehen sind. Wird eine bestimmte Sammlung an Gefäßformen in einem Findspot gefunden, in einem anderen jedoch nicht, so gibt dies Rückschlüsse auf die Datierung des Findspots. Die nachfolgenden Beispiele geben einen kleinen Einblick in eine mögliche Zeitenmodellierung der Samian Ware (Terra Sigillata) in Kombination mit den Allen Zeitrelationen und zeigt mögliche Probleme auf.

Möchte man z.B. Rückschlüsse auf die Zeit der Nutzung einer bestimmten Gefäßform[527] ziehen, kann man wie folgt vorgehen: Zunächst wird die Datenbank nach Sites durchsucht, in denen eine bestimmte Gefäßform[528] vorkommt (z.B. R1, R2 und R3, vgl. Abbildung 9-22). Diese Sites werden nach gleichen zeitlichen Ausdehnungen untersucht und z.B. bei ähnlichen geographischen Koordinaten oder anderen Faktoren (Kastellketten etc.) zu einem Cluster (z.B. Cluster A) zusammenge-

[527] siehe auch http://potsherd.net/atlas/types/sigillata (abgerufen 04.12.2013)

[528] siehe auch http://de.wikipedia.org/wiki/Liste_wichtiger_Terra-Sigillata-Gef%C3%A4%C3%9Fformen (abgerufen 04.12.2013)

fasst. Dieses Clustering kann in den Allen-Relationen mit einem equals (=) ausgedrückt werden. Hierbei ist jedoch zu beachten, dass die Cluster nur als virtuelle Elemente dienen. In der Realität würde man z.B. in Cluster C auf die Site C1 referenzieren, die durch die equals Relationen alle anderen Elemente des Clusters einschließt. Nach diesem Verfahren entstehen viele Cluster mit einem Vorkommen einer bestimmen Gefäßform, bzw. mehreren Gefäßformen. Diese Cluster können nun ebenfalls mit Allen-Zeitrelationen in Verbindung gesetzt werden und somit eine relativ-chronologische Abbildung der Findspots bzw. Sites, erzeugt werden. Möchte man die beispielsweise die Zeitspanne der Gefäßform R1 erhalten, so ist nach einer Site zu suchen, in der die Gefäßform R1 noch angewendet wurde. In diesem Beispiel ist dies in Cluster C der Fall. Cluster D enthält nur noch die Gefäßform R2, so dass die Gefäßform R1 mit Beginn des Clusters D endet. Die Gefäßform R2 beginnt jedoch mit Cluster C und endet mit Cluster D, da diese sie nicht in Cluster A und B oder in Cluster E vorkommt. Abbildung 9-22 zeigt somit, dass die Nutzung der Gefäßform R1 der Allen Relation R1 starts Cluster C, R2 der Relation R2 started by Cluster C bzw. finished by Cluster D und R3 started by Cluster E, entsprechen kann. Auf ein konkretes Beispiel wird aufgrund der aggregierten Datenlage verzichtet.

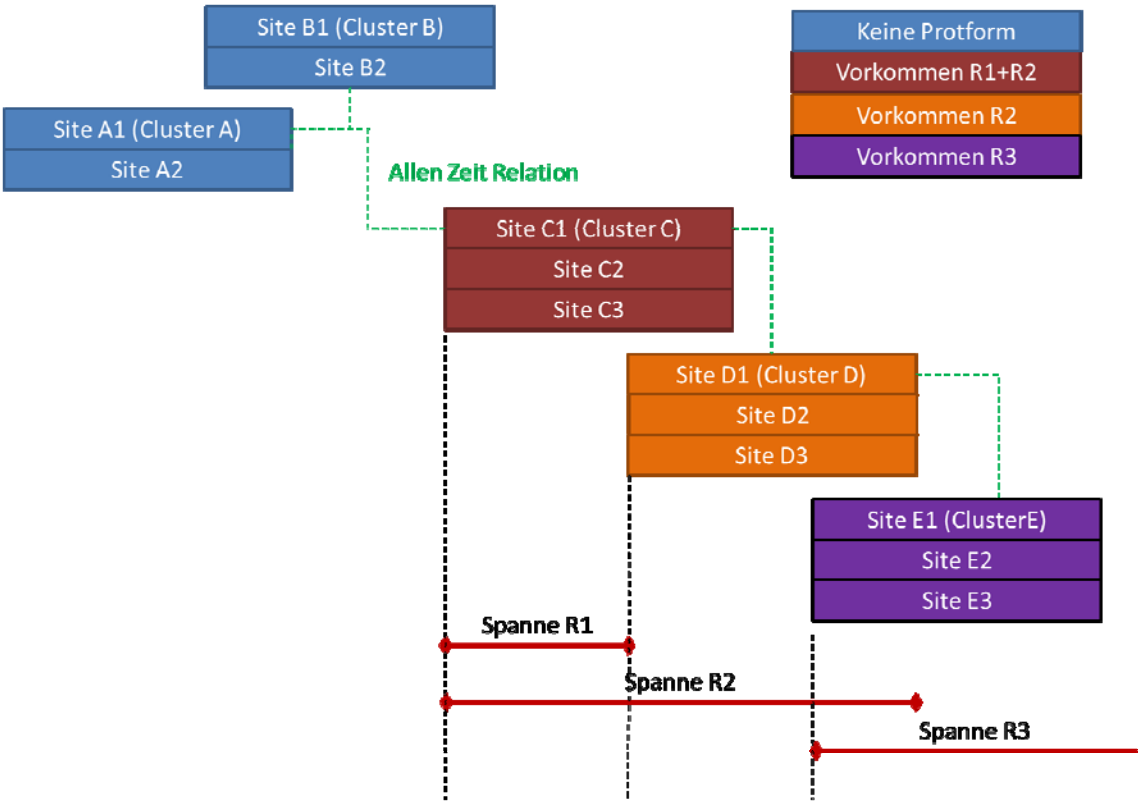


Abbildung 9-22: Site-Clustering nach Gefäßformen

Es stellt sich zudem Frage, welche zeitlichen Relationen ein Findspot, Potter und ein Fragment untereinander eingehen können. Die Erstellung eines Fragments ist direkt an einen Töpfer gebunden. Das Fragment kann während der Lebenszeit eines Töpfers hergestellt worden sein (during), mit dem Tod eines Töpfers zerstört (finishes) oder über den Tod des Töpfers hinaus weiterverhandelt werden (overlapped by). Insbesondere die erste und dritte Variante können als realistisch be-

trachtet werden. Betrachtet man die zeitlichen Relationen eines Fragments zu einem Findspot ist dieses bei weitem schwieriger. Ein Fragment kann in der Zeit des aktiven Lebens in einem Findspot hergestellt und verhandelt worden sein, jedoch auch von einem anderen Ort (z.B. durch Feldzüge) in einen neuen Findspot überführt und damit auch von einem Findspot zu einem anderen Ort gebracht werden. Dies bedeutet, dass alle Allen-Relationen zwischen einem Fragment und einem Site möglich sind, gleiches gilt für einen Töpfer und einer Site, der gereist sein kann. Eindeutiger ist es nur, wenn ein Fragment in einer Site gefunden wurde, so entfallen die Relationen after, met by, overlapped by, contains und started by, da das Fragment nicht über die Lebensdauer einer bestimmten Site existiert haben kann (siehe auch Abbildung 9-23).

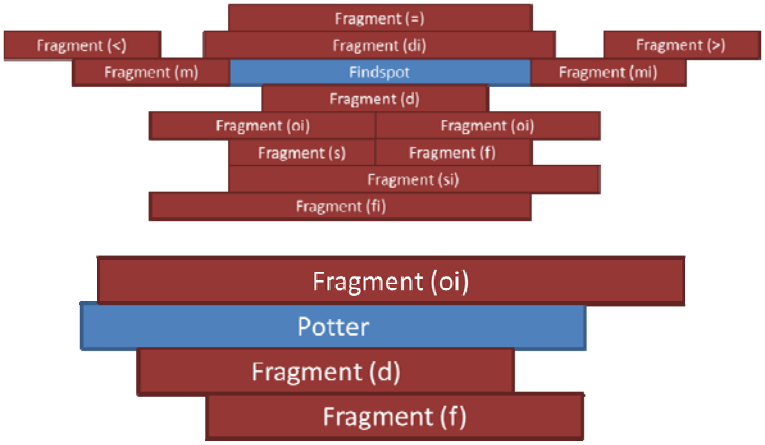


Abbildung 9-23: Allen Relationen von Potter und Findspot zu Fragment

Mit diesen Informationen kann an einem kleinen Beispiel exemplarisch die Komplexität der Allen Relationen und deren Schlussfolgerungen im Kontext von Töpfern, Fragmenten, Gefäßformen und Sites gezeigt werden (vgl. Abbildung 9-24). Es seien gegeben: drei verschiedene Fragmente mit unterschiedlichen Gefäßformen (X,Y,Z), zwei Töpfer (A,B) und drei verschiedene Sites (X/Y, Y/Z, X/Z). Töpfer A hat sowohl die Gefäßform X, wie auch Y genutzt. Töpfer B nutzt nur Gefäßform B. Diese Gefäßformen wurden in verschiedenen Sites gefunden, z.B. Potform X und Z in Site X/Z. Darüber hinaus können den Fragmenten die zuvor gezeigten Relationen zu einem Töpfer zugewiesen werden. Die Beziehung zu einer Site ist jedoch, wie bereits beschrieben, sehr unklar. Diese Ausgangsstellung lässt schlussfolgern, dass eine Ableitung von Beziehungen schwierig ist. So lässt sich sagen, dass das Vorkommen der Gefäßformen X und Z in Site X/Z keine direkt ersichtliche Auswirkung auf die beteiligten Töpfer oder Fragmente hat. Dies könnte durch das Einführen eines Datums an allen drei markanten Punkten vereinfacht werden. Diese Untersuchungen der Auswirkungen von Datumszusätzen sollten in weiterführenden Studien umgesetzt werden.

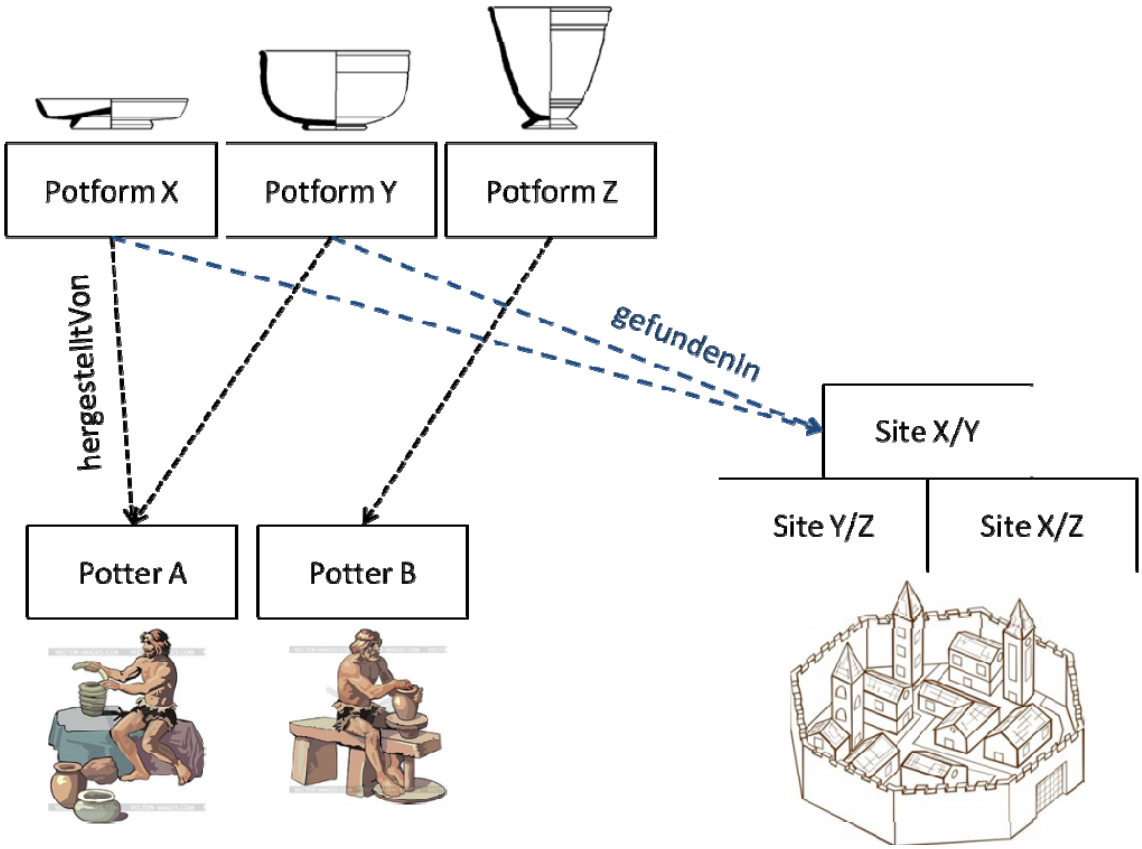


Abbildung 9-24: Allen Relationen in der Samian Ware (Beispiel)

Zudem stellt sich die Frage nach den Auswirkungen von chronologischen Beziehungen zwischen einer Site, in der beispielsweise zwei Gefäßformen gefunden wurden und einer Site in der eine andere Gefäßform vermutet wird (vgl. Abbildung 9-25). Besteht z.B. die Relation Site X/Y overlaps Site Z, so heißt dies, dass Site Z overlapped by Site X/Y ist. Dies entspricht der Inverse der Relation, welche im Modell nach Allen mit einem i gekennzeichnet sind. Somit gilt Site A Relation Site B entspricht Site B Relation(Inverse) Site A.

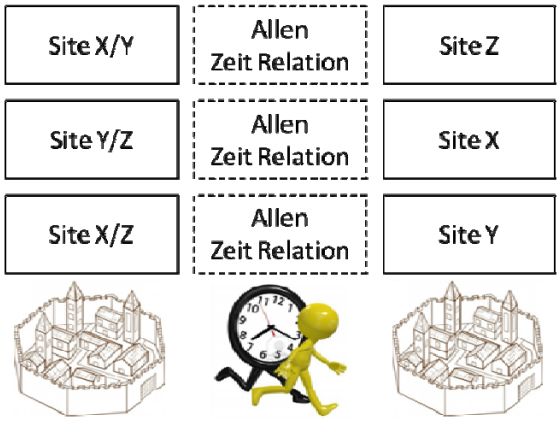


Abbildung 9-25: Allen Relationen Site X/Y zu Site Z

Die gezeigten Beispiele verdeutlichen, dass eine Kombination mit der in Kapitel 9.4 dargestellten Simulation in relativen Chronologien sinnvoll ist. Im Vergleich zu den im Time Explorer genutzten einfachen Beispielen, ist die Chronologie in den Daten als wesentlich komplexer anzusehen. Erst nach einer Aufbereitung dieser Daten in eine relative Chronologie können mit der entwickelten Ontologie und dem Time Explorer Simulationen erzeugt werden. Hier ist noch ein weiterer Entwicklungsschritt zu gehen und Entwicklungszeit zu investieren.

10 Fazit und Ausblick

Die hier vorliegende Masterarbeit zeigt insbesondere die Problematik der semantischen Modellierung archäologischer Fachdaten (hier Töpfer, Fund- und Produktionsstätten und Terra Sigillata Fragmente) und deren Migration in semantische Formate (XML, JSON, RDF), wie Linked Open Data (RDF). Die Beschreibung der Attribute und der Beziehungen zwischen einzelnen Ressourcen wird mit bereits bestehenden LOD-Konzepten, kontrollierten Vokabularien und eigenen Lösungen umgesetzt. Des Weiteren wird die Bereitstellung der modellierten Daten als LOD (in einer ReST-Schnittstelle und im Sesame Triplestore), sowie in OGC konformen Webdiensten exemplarisch dargestellt. Insbesondere die Verlinkung zu anderen bereits bestehenden Projekten und deren Nutzen, wird am Beispiel von Pleiades und Pelagios erläutert. Darüber hinaus kann das Potential der Verlinkungen und Abfragen von heterogenen Informationen zwischen Töpfern, Fragmenten und Orten und deren relativ-chronologische Beziehungen über ein Tool aufgezeigt werden. Die Komplexität des Netzes sich bedingender Datierungen der Funde ist im Datenmodell direkt abgebildet, so dass das System schnell auf Modifikationen reagiert und eine Echtzeitvisualisierung der Simulation von Änderungen in den Beziehungen möglich ist.

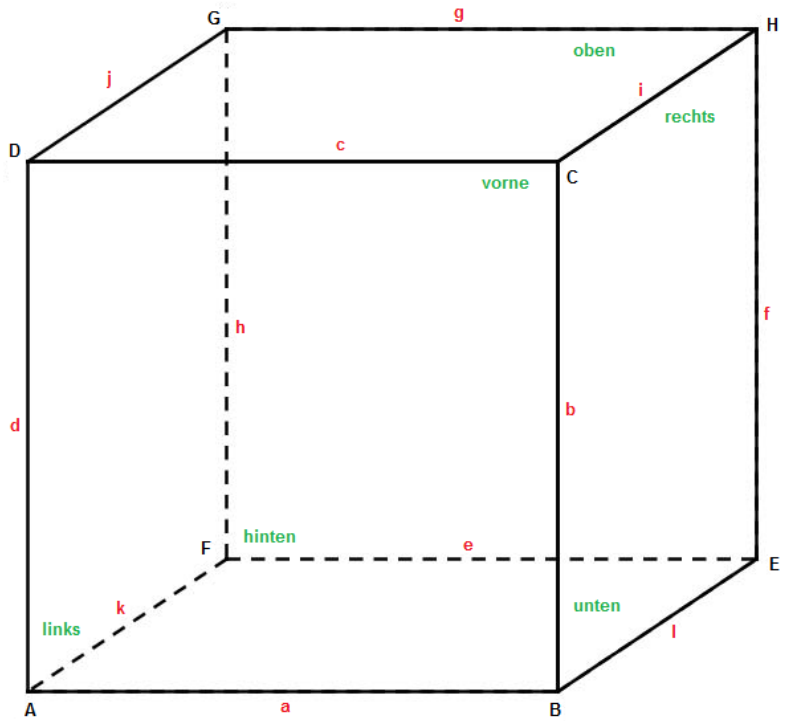
Die Ergebnisse dieser Arbeit zeigen, dass Forschungen im Bereich der römischen Keramik vom Prinzip der Linked Open Data profitieren. Exemplarisch wird in der Masterarbeit auf die südgallische Terra Sigillata eingegangen. Das Prinzip lässt sich jedoch auf viele weitere Forschungsgebiete in der Archäologie übertragen. Forschung in diesen Gebieten kann, angereichert durch neues Wissen, neue Erkenntnisse erlangen. Die entwickelte PelagiosConnectionAPI ermöglicht hierbei den Zugriff auf große LOD-Repositorien, die in Verbindung mit den eigenen semantisch modellierten Daten ein großes Arsenal an Erkundigungsmaterialien bereitstellen. Vielversprechende Perspektiven entstehen hierbei vor allem durch die Verknüpfung mit weiteren Ressourcen wie Matrizen, Formschüsseln und Münzen.

Die entwickelten Applikationen zur semantischen Datenbereitstellung (ReST-Schnittstelle), grafischen Visualisierung der Fund- und Produktionsstädten (Map), sowie der Verlinkungen zwischen Töpfern, Keramikfragmenten und jenen Orten (Linked Samian Ware Explorer) und die Simulation von relativ chronologischen Änderungen mittels RDF (Time Explorer), zeigen das zuvor erwähnte Potential auf.

Insbesondere die semantische Modellierung der drei Objektarten (historische Personen, archäologische Objekte und antike Orte) kann als problematisch angesehen werden. Bereits verfügbare Standards und Lösungsvorschläge andere Projekte bilden nicht exakt die Relationen der hier genutzten Terra Sigillata Daten und deren Beziehungen untereinander wieder. Im Bereich der XML-Modellierung könnte dies mit MIDAS XML gelöst werden. Die Bereitstellung eines WFS und damit GML und GeoJSON, erzeugen Interoperabilität, besonders in den Bereichen der Archäologie, wel-

che auf GIS Analysen basieren. Konzepte wie Pleiades oder FOAF sind zweckmäßig, sollten jedoch durch ein Gesamtkonzept der drei Objektarten ersetzt werden. Eines dieser Konzepte könnte CIDOC CRM sein. Ein Mapping der Samian Ware Daten nach CIDOC CRM ist Arbeits- und Zeitaufwändig. Dem gegenüber steht die Errichtung einer eigenen Ontologie, welche den gesamten Prozess der Terra Sigillata (Produktion, Verhandlung, Verstörung) und alle Attribute (z.B. Potformen, Töpfermatrizen, etc.) abbildet. Des Weiteren ist die Verlinkung zwischen den modellierten Ressourcen problematisch. Das genutzte Konzept der Open Annotation ermöglicht eine implizite Beziehung ohne explizite Angabe eines Prädikats, lässt ein System jedoch über die genaue Beziehung im Unklaren. Dies könnte ebenfalls durch ein kontrolliertes Vokabular der Beziehungen gelöst werden. Die Datenbereitstellung erfolgt über eine JAVA-ReST-Schnittstelle mit den Frameworks JAXB und Jersey, sowie den Geoserver und einen Sesame Triplestore. Die Modellierung erfolgt hierbei über einfache Zeichenketten (String), eine Objektmodellierung in JAVA ist nicht vorgesehen. Virtuoso Server bieten die Möglichkeit z.B. PostGIS-Datenbanken einzubinden und somit eine einheitliche Veröffentlichung von XML, JSON und Linked Data. Dem Virtuoso Server ist ebenfalls ein Triplestore zur Befragung mit SPARQL integriert. Kommende Projekte sollten diese oder eine anderes Konzept verwenden, sodass semantische Modellierungen einer Ontologie als Objekte zur Verfügung gestellt werden können. Im Linked Data Explorer sind die Möglichkeiten der Abfrage von LOD dargestellt. Durch ein Mapping zu anderen großen LOD Projekten (hier Pleiades und Pelagios) ist der Erhalt ergänzender Information (hier z.B. zu einem Fundort) möglich. Dieses Mapping ist jedoch ein manueller Prozess und mit Schwierigkeiten verbunden, da unterschiedliche Wissenschaftler und verschiedene Fachdisziplinen Sachverhalte unterschiedlich auslegen. Dem Geisteswissenschaftler wird hier ein Tool zur Verfügung gestellt, das räumliche Abfragen von Orten eines anderen Repositoriums zu den eigenen Orten tätigt und so die räumlich nächsten Orte ermittelt. Dies kann jedoch nur als Hilfestellung gesehen werden. Auf Grund des Zeitaufwands ist ein (halb-) automatisiertes Mapping anzustreben. Der Aspekt der relativen Chronologie bietet viele Möglichkeiten der weitergehenden Forschung an. In dieser Arbeit werden prototypisch eine relativ chronologische Ontologie nach [ALLEN, 1992] und eine Simulationsplattform erstellt. Sie zeigen Potentiale des Übergangs von absolut chronologischen Daten zu relativen Chronologien auf. Hier sind insbesondere der Aspekt der Visualisierung (Timeline, Graph, Text, etc.) und die Weiterentwicklung der Ontologie inkl. Regeln als weiterführende Forschungsvorhaben zu nennen. Die komplexe Struktur der Datierung in den Terra Sigillata Daten muss zudem neu aufgeschlüsselt (keine Aggregationen, wie hier am Beispiel der Findspots) werden. Nur anhand der nicht aggregierten relativ chronologischen Beziehungen ist eine Simulation mit realen Daten möglich.

Anhang A: RDF Beispiel Cube



Anhang 1: Modellierung eines Cibes (Beispiel)

```
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix xsd:<http://www.w3.org/2001/XMLSchema#>.
@prefix cube:<http://www.cubes.com/>.
@prefix punkt:<http://www.cubeelements.com/punkt/>.
@prefix linie:<http://www.cubeelements.com/linie/>.
@prefix flaeche:<http://www.cubeelements.com/flaeche/>.
@prefix voc:<http://www.voc.de/>.
#Punkte
punkt:A voc:cube cube:cube1 . punkt:B voc:cube cube:cube1 .
punkt:C voc:cube cube:cube1 . punkt:D voc:cube cube:cube1 .
punkt:E voc:cube cube:cube1 . punkt:F voc:cube cube:cube1 .
punkt:G voc:cube cube:cube1 .punkt:H voc:cube cube:cube1 .
#Linien
punkt:A voc:line linie:a . punkt:B voc:line linie:a . punkt:B voc:line linie:b .
punkt:C voc:line linie:b . punkt:C voc:line linie:c . punkt:D voc:line linie:c .
```

```
punkt:D voc:line linie:d . punkt:A voc:line linie:d . punkt:F voc:line linie:e .
punkt:E voc:line linie:e . punkt:E voc:line linie:f . punkt:H voc:line linie:f .
punkt:G voc:line linie:g . punkt:H voc:line linie:g . punkt:G voc:line linie:h .
punkt:F voc:line linie:h . punkt:C voc:line linie:i . punkt:H voc:line linie:i .
punkt:D voc:line linie:j . punkt:G voc:line linie:j . punkt:A voc:line linie:k .
punkt:F voc:line linie:k . punkt:B voc:line linie:l . punkt:E voc:line linie:l .
#Flaechen
punkt:A voc:areap flaeche:vorne . punkt:A voc:areap flaeche:vorne .
punkt:B voc:areap flaeche:vorne . punkt:C voc:areap flaeche:vorne .
punkt:D voc:areap flaeche:vorne . punkt:B voc:areap flaeche:rechts .
punkt:C voc:areap flaeche:rechts . punkt:H voc:areap flaeche:rechts .
punkt:E voc:areap flaeche:rechts . punkt:G voc:areap flaeche:hinten .
punkt:H voc:areap flaeche:hinten . punkt:F voc:areap flaeche:hinten .
punkt:E voc:areap flaeche:hinten . punkt:G voc:areap flaeche:links .
punkt:F voc:areap flaeche:links . punkt:D voc:areap flaeche:links .
punkt:A voc:areap flaeche:links . punkt:A voc:areap flaeche:unten .
punkt:B voc:areap flaeche:unten . punkt:F voc:areap flaeche:unten .
punkt:E voc:areap flaeche:unten . punkt:D voc:areap flaeche:oben .
punkt:C voc:areap flaeche:oben . punkt:H voc:areap flaeche:oben .
punkt:G voc:areap flaeche:oben . linie:a voc:areal flaeche:vorne .
linie:b voc:areal flaeche:vorne . linie:c voc:areal flaeche:vorne .
linie:d voc:areal flaeche:vorne . linie:b voc:areal flaeche:rechts .
linie:i voc:areal flaeche:rechts . linie:f voc:areal flaeche:rechts .
linie:l voc:areal flaeche:rechts . linie:e voc:areal flaeche:hinten .
linie:f voc:areal flaeche:hinten . linie:g voc:areal flaeche:hinten .
linie:h voc:areal flaeche:hinten . linie:k voc:areal flaeche:links .
linie:d voc:areal flaeche:links . linie:j voc:areal flaeche:links .
linie:h voc:areal flaeche:links . linie:a voc:areal flaeche:unten .
linie:l voc:areal flaeche:unten . linie:e voc:areal flaeche:unten .
linie:k voc:areal flaeche:unten . linie:c voc:areal flaeche:oben .
linie:i voc:areal flaeche:oben . linie:g voc:areal flaeche:oben .
linie:j voc:areal flaeche:oben .
```

Punkt A von Punkt H ermitteln

```
PREFIX voc:<http://www.voc.de/>
PREFIX punkt:<http://www.cubeelements.com/punkt/>
PREFIX linie:<http://www.cubeelements.com/linie/>

SELECT DISTINCT ?p
WHERE {
   punkt:A voc:line ?l1.
   ?x voc:line ?l1.
   FILTER(punkt:A != ?x)
   ?x voc:line ?l2.
   ?z voc:line ?l2.
   FILTER(?x != ?z)
   FILTER(?l1 != ?l2)

   punkt:A voc:line ?l3.
   ?y voc:line ?l3.
   FILTER(punkt:A != ?y)
   ?y voc:line ?l4.
   ?z voc:line ?l4.
   FILTER(?y != ?z)
   FILTER(?l3 != ?l4)

   FILTER(?x != ?y)

   ?z voc:line ?l5.
   ?p voc:line ?l5.
   FILTER(?z != ?p)
   FILTER(?l5 != ?l4)
   FILTER(?l5 != ?l2)
}
```

Anhang B: Installationshinweise Server

Erzeugung der PostGIS Datenbankstruktur mittels PuTTY und Commandline[529].

Change to root user

login as: **thiery**

Root → **thiery@geinarfa$ sudo su**

Datenbank samian mit POSTGIS Extension erstellen

psql Verzeichnis → **root@geinarfa# cd usr/pgsql-9.2/bin**

User Postgres → **root@geinarfa# su postgres**

Create Database: **bash CREATEDB samian**

Login Database → **bash psql samian**

Enable Postgis → **samian=# CREATE EXTENSION postgis;**

Passwort des user Postgres ändern:

root@geinarfa# sudo -u postgres psql

postgres=# \password postgres

Zugriff auf Datenbank samian:

root@geinarfa# psql -U postgres -d samian

Daten aus sql Script in Datenbank kopieren (Beispiel Create Table / Input and Create Geometry)

samian# SET client_encoding to "UTF-8";

samian# \i 'C:/temp/CreateInsert.sql'

SQL Copy CSV to Postgres

SET client_encoding = 'win1252';

COPY <tabellenname> FROM '<dateipfad>' WITH DELIMITER AS ';';

SELECT AddGeometryColumn('<tabellenname>', 'geom', 4326, 'POINT', 2);

UPDATE <tabellenname>SET geom = ST_SetSRID(ST_Point(lon, lat),4326);

---

[529] Tutorials siehe http://postgis.net/install und http://www.if-not-true-then-false.com/2012/install-postgresql-on-fedora-centos-red-hat-rhel

Anhang C: JAVA-Quellcode

Der JAVA Quellcode befindet auf dem Datenträger.

Anhang D: Kilnsites - Karten und Ausgangsdaten

Die Karten und Ausgangsdaten befinden auf dem Datenträger.

Anhang E: SQL-Skripte

Die SQL-Skripte befinden sich auf dem Datenträger.

Anhang F: Eigene Ontologie

```
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix sam:   <http://samianontology.net/historical> .
@prefix pred:  <http://samianontology.net/historical/pred> .

# SKOS Referenzen
sam:Person rdf:type skos:Concept;
  skos:prefLabel "Personen"@de;
  skos:definition "Historische und moderne Personen"@de .
sam:Place rdf:type skos:Concept;
  skos:prefLabel "Places"@de;
  skos:definition "Historische und Moderne Orte"@de .
sam:ArchaeologicalData rdf:type skos:Concept;
  skos:prefLabel "Archäologische Objekte"@de;
  skos:definition "Archäologische Objekte"@de .
sam:TimePeriod rdf:type skos:Concept;
  skos:prefLabel "Zeitperioden"@de;
  skos:definition "Zeitperioden"@de .

# Potter (Person) – Verzicht auf rdfs:Label und rdfs:Comment (Bsp. in Property)
sam:Person                rdf:type          rdfs:Class .
sam:Person                rdfs:SubClassOf   rdfs:Ressource .
sam:historicalPerson      rdf:type          rdfs:Class .
sam:historicalPerson      rdfs:SubClassOf   rdfs:Person .
sam:modernPerson          rdf:type          rdfs:Class .
sam:modernPerson          rdfs:SubClassOf   rdfs:Person .
sam:PersonData            rdf:type          rdfs:Class .
sam:PersonData            rdfs:SubClassOf   rdfs:Person .
# historcal Persons
sam:Potter                rdf:type          rdfs:Class .
sam:Potter                rdf:SubClassOf    rdfs:historicalPerson .
sam:Regent                rdf:type          rdfs:Class .
```

```
sam:Regent              rdf:SubClassOf    rdfs:historicalPerson .
sam:Citizen             rdf:type          rdfs:Class .
sam:Citizen             rdf:SubClassOf    rdfs:historicalPerson .
sam:Slave               rdf:type          rdfs:Class .
sam:Slave               rdf:SubClassOf    rdfs:historicalPerson .
# modern Persons
sam:Student             rdf:type          rdfs:Class .
sam:Student             rdf:SubClassOf    rdfs:modernPerson .
sam:ITdeveloper         rdf:type          rdfs:Class .
sam:ITdeveloper         rdf:SubClassOf    rdfs:modernPerson .
sam:Archaelogist        rdf:type          rdfs:Class .
sam:Archaelogist        rdf:SubClassOf    rdfs:modernPerson .
# Data
sam:Sex                 rdf:type          rdfs:Class .
sam:Sex                 rdf:SubClassOf    rdfs:PersonData .
sam:Male                rdf:type          rdfs:Class .
sam:Male                rdf:SubClassOf    rdfs:Sex .
sam:Female              rdf:type          rdfs:Class .
sam:Female              rdf:SubClassOf    rdfs:Sex .

# Findspot (Ort)
sam:Place               rdf:type          rdfs:Class .
sam:Place               rdfs:SubClassOf   rdfs:Ressource .
sam:historicalPlace     rdf:type          rdfs:Class .
sam:historicalPlace     rdfs:SubClassOf   rdfs:Place .
sam:modernPlace         rdf:type          rdfs:Class .
sam:modernPlace         rdfs:SubClassOf   rdfs:Place .
sam:PlaceData           rdf:type          rdfs:Class .
sam:PlaceData           rdfs:SubClassOf   rdfs:Place .
# historical Place
sam:Site                rdf:type          rdfs:Class .
sam:Site                rdf:SubClassOf    rdfs:historicalPlace .
sam:Findspot            rdf:type          rdfs:Class .
sam:Findspot            rdf:SubClassOf    rdfs:Site .
# modern Place
sam:Country             rdf:type          rdfs:Class .
sam:Country             rdf:SubClassOf    rdfs:modernPlace .
sam:Region              rdf:type          rdfs:Class .
sam:Region              rdf:SubClassOf    rdfs:Country .
sam:City                rdf:type          rdfs:Class .
sam:City                rdf:SubClassOf    rdfs:Region .
```

```
# Data
sam:PlaceTypes                rdf:type          rdfs:Class .
sam:PlaceTypes                rdf:SubClassOf    rdfs:PlaceData .
sam:ProductionCentre          rdf:type          rdfs:Class .
sam:ProductionCentre          rdf:SubClassOf    rdfs:PlaceTypes .
sam:Settlement                rdf:type          rdfs:Class .
sam:Settlement                rdf:SubClassOf    rdfs:PlaceTypes .
sam:Findspot                  rdf:type          rdfs:Class .
sam:Findspot                  rdf:SubClassOf    rdfs:PlaceTypes .
sam:PotteryProductionCentre   rdf:type          rdfs:Class .
sam:PotteryProductionCentre   rdf:SubClassOf    rdfs:ProductionCentre .

# Fragmente
sam:ArchaeologicalData        rdf:type          rdfs:Class .
sam:ArchaeologicalData        rdfs:SubClassOf   rdfs:Ressource .
sam:Pottery                   rdf:type          rdfs:Class .
sam:Pottery                   rdfs:SubClassOf   rdfs:ArchaeologicalData .
sam:Coins                     rdf:type          rdfs:Class .
sam:Coins                     rdfs:SubClassOf   rdfs:ArchaeologicalData .
sam:Bones                     rdf:type          rdfs:Class .
sam:Bones                     rdfs:SubClassOf   rdfs:ArchaeologicalData .
sam:SamianWare                rdf:type          rdfs:Class .
sam:SamianWare                rdfs:SubClassOf   rdfs:Pottery .
sam:RomanCoins                rdf:type          rdfs:Class .
sam:RomanCoins                rdfs:SubClassOf   rdfs:Coins .
sam:GreekCoins                rdf:type          rdfs:Class .
sam:GreekCoins                rdfs:SubClassOf   rdfs:Coins .
sam:HumanBone                 rdf:type          rdfs:Class .
sam:HumanBone                 rdfs:SubClassOf   rdfs:Bones .
sam:AnimalBone                rdf:type          rdfs:Class .
sam:AnimalBone                rdfs:SubClassOf   rdfs:Bones .

# Prädikatenhierarchie
pred:ID                       rdf:type              rdfs:Property .
pred:ID                       rdfs:domain           sam:Person .
pred:ID                       rdfs:domain           sam:Place .
pred:ID                       rdfs:domain           sam:ArchaeologicalData .
pred:Name                     rdfs:SubPropertyOf    pred:ID .
pred:Name                     rdfs:label            "Name"@de .
pred:Name                     rdfs:comment          "Names"@de .
```

```
pred:Name             rdfs:domain           sam:Place .
pred:Name             rdfs:domain           sam:Person .
pred:Name             rdfs:range            xsd:string .
pred:PersonName       rdfs:SubPropertyOf    pred:Name .
pred:PersonName       rdfs:domain           sam:Person .
pred:PersonName       rdfs:range            xsd:string .
pred:PlaceName        rdfs:SubPropertyOf    pred:Name .
pred:PlaceName        rdfs:domain           sam:Place .
pred:PlaceName        rdfs:range            xsd:string .
pred:Job              rdf:type              rdfs:Property .
pred:Job              rdfs:domain           sam:Person .
pred:Job              rdfs:range            xsd:string .
pred:TimePeriod       rdf:type              rdfs:Property .
pred:TimePeriod       rdfs:domain           sam:TimePeriod .
pred:TimePeriod       rdfs:range            sam:TimePeriod .
pred:Sex              rdf:type              rdfs:Property .
pred:Sex              rdfs:domain           sam:Person .
pred:Sex              rdfs:range            sam:Sex .
pred:PlaceType        rdf:type              rdfs:Property .
pred:PlaceType        rdfs:domain           sam:PlaceTypes .
pred:PlaceType        rdfs:range            sam:PlaceTypes .
pred:Location         rdf:type              rdfs:Property .
pred:Location         rdfs:domain           sam:Place .
pred:Location         rdfs:range            sam:Place .
pred:WGS84            rdfs:SubPropertyOf    pred:Location .
pred:WGS84            rdfs:domain           sam:Place .
pred:Lat              rdf:SubPropertyOf     pred:WGS84 .
pred:Lat              rdfs:domain           sam:Place .
pred:Lat              rdfs:range            xsd:double .
pred:Lon              rdf:SubPropertyOf     pred:WGS84 .
pred:Lon              rdfs:domain           sam:Place .
pred:Lon              rdfs:range            xsd:double .
pred:UTM32            rdfs:SubPropertyOf    pred:Location .
pred:UTM32            rdfs:domain           sam:Place .
pred:North            rdf:SubPropertyOf     pred:UTM32 .
pred:North            rdfs:domain           sam:Place .
pred:North            rdfs:range            xsd:double .
pred:East             rdf:SubPropertyOf     pred:UTM32 .
pred:East             rdfs:domain           sam:Place .
pred:East             rdfs:range            xsd:double .
```

```
pred:Action           rdf:type              rdfs:Property .
pred:IsStampedFrom    rdf:SubPropertyOf     rdfs:Action .
pred:IsStampedFrom    rdfs:domain           sam:historicalPerson .
pred:IsStampedFrom    rdfs:range            sam:historicalPerson .
pred:WasFoundIn       rdf:SubPropertyOf     pred:Action .
pred:WasFoundIn       rdfs:domain           sam:Place .
pred:WasFoundIn       rdfs:range            sam:Place .
pred:HasStamped       rdf:SubPropertyOf     pred:Action .
pred:HasStamped       rdfs:domain           sam:Pottery .
pred:HasStamped       rdfs:range            sam:Pottery .
pred:Property         rdf:type              rdfs:Property .
pred:Property         rdfs:domain           sam:Pottery .
pred:Property         rdfs:range            sam:Pottery .
pred:SamianWareProp   rdf:SubPropertyOf     pred:Property .
pred:SamianWareProp   rdfs:domain           sam:Pottery .
pred:SamianWareProp   rdfs:range            sam:Pottery .
pred:Potform          rdfs:SubPropertyOf    pred:SamianWareProp .
pred:Potform          rdfs:domain           sam:SamianWare .
pred:Potform          rdfs:range            xsd:string .
pred:Die              rdfs:SubPropertyOf    pred:SamianWareProp .
pred:Die              rdfs:domain           sam:SamianWare .
pred:Die              rdfs:range            xsd:string .
pred:Number           rdfs:SubPropertyOf    pred:SamianWareProp .
pred:Number           rdfs:domain           sam:SamianWare .
pred:Number           rdfs:range            xsd:integer .
```

Anhang G: SLD der Kilnsites

Das SLD der Kilnsites befindet sich auf dem Datenträger.

Anhang H: Liste der Mapping Ergebnisse

```
INSERT INTO pleiadesmatch VALUES ('Baldock', '79311');
INSERT INTO pleiadesmatch VALUES ('Ribchester', '79352');
INSERT INTO pleiadesmatch VALUES ('Caersws', '79363');
INSERT INTO pleiadesmatch VALUES ('Silchester', '79368');
INSERT INTO pleiadesmatch VALUES ('Castell Collen', '79378');
INSERT INTO pleiadesmatch VALUES ('Camulodunum', '79393');
INSERT INTO pleiadesmatch VALUES ('Colchester', '79393');
INSERT INTO pleiadesmatch VALUES ('Lincoln', '79395');
INSERT INTO pleiadesmatch VALUES ('Northwich', '79400');
INSERT INTO pleiadesmatch VALUES ('Little Chester', '79419');
INSERT INTO pleiadesmatch VALUES ('Chester', '79420');
INSERT INTO pleiadesmatch VALUES ('Canterbury', '79439');
INSERT INTO pleiadesmatch VALUES ('Caerleon', '79532');
INSERT INTO pleiadesmatch VALUES ('Caerhun', '79542');
INSERT INTO pleiadesmatch VALUES ('Cirencester', '79549');
INSERT INTO pleiadesmatch VALUES ('Castleford', '79553');
INSERT INTO pleiadesmatch VALUES ('London', '79574');
INSERT INTO pleiadesmatch VALUES ('Carmarthen', '79602');
INSERT INTO pleiadesmatch VALUES ('Chichester', '79622');
INSERT INTO pleiadesmatch VALUES ('Richborough', '79664');
INSERT INTO pleiadesmatch VALUES ('Cardiff', '79706');
INSERT INTO pleiadesmatch VALUES ('Caerwent', '79733');
INSERT INTO pleiadesmatch VALUES ('Caistor-by-Norwich', '79735');
INSERT INTO pleiadesmatch VALUES ('Wilderspool', '79763');
INSERT INTO pleiadesmatch VALUES ('Melandra', '79777');
INSERT INTO pleiadesmatch VALUES ('Great Chesters', '89095');
INSERT INTO pleiadesmatch VALUES ('Watercrook', '89102');
INSERT INTO pleiadesmatch VALUES ('South Shields', '89104');
INSERT INTO pleiadesmatch VALUES ('Birrens', '89117');
INSERT INTO pleiadesmatch VALUES ('Carrawburgh', '89128');
INSERT INTO pleiadesmatch VALUES ('Catterick', '89143');
```

```
INSERT INTO pleiadesmatch VALUES ('Chesters', '89144');
INSERT INTO pleiadesmatch VALUES ('Camelon', '89148');
INSERT INTO pleiadesmatch VALUES ('Condercum', '89150');
INSERT INTO pleiadesmatch VALUES ('Corbridge', '89152');
INSERT INTO pleiadesmatch VALUES ('Carlisle', '89234');
INSERT INTO pleiadesmatch VALUES ('Stanwix', '89234');
INSERT INTO pleiadesmatch VALUES ('Newcastle', '89275');
INSERT INTO pleiadesmatch VALUES ('Strageath', '89293');
INSERT INTO pleiadesmatch VALUES ('Chesterholm-Vindolanda', '89313');
INSERT INTO pleiadesmatch VALUES ('Bainbridge', '89317');
INSERT INTO pleiadesmatch VALUES ('Old Penrith', '89318');
INSERT INTO pleiadesmatch VALUES ('Woerden', '98997');
INSERT INTO pleiadesmatch VALUES ('Valkenburg', '99021');
INSERT INTO pleiadesmatch VALUES ('Utrecht', '99046');
INSERT INTO pleiadesmatch VALUES ('Velsen', '99058');
INSERT INTO pleiadesmatch VALUES ('Aardenburg', '108722');
INSERT INTO pleiadesmatch VALUES ('Baden-Baden', '108748');
INSERT INTO pleiadesmatch VALUES ('Bendorf', '108804');
INSERT INTO pleiadesmatch VALUES ('Blickweiler', '108814');
INSERT INTO pleiadesmatch VALUES ('Bliesbruck', '108816');
INSERT INTO pleiadesmatch VALUES ('Butzbach', '108839');
INSERT INTO pleiadesmatch VALUES ('Eincheville', '108888');
INSERT INTO pleiadesmatch VALUES ('Trier', '108894');
INSERT INTO pleiadesmatch VALUES ('Eschweilerhof', '108966');
INSERT INTO pleiadesmatch VALUES ('Horbourg', '109053');
INSERT INTO pleiadesmatch VALUES ('Langenhain', '109100');
INSERT INTO pleiadesmatch VALUES ('Lavoye', '109103');
INSERT INTO pleiadesmatch VALUES ('La Madeleine', '109131');
INSERT INTO pleiadesmatch VALUES ('Mainz', '109169');
INSERT INTO pleiadesmatch VALUES ('Heddernheim', '109205');
INSERT INTO pleiadesmatch VALUES ('Saalburg', '109297');
INSERT INTO pleiadesmatch VALUES ('Sinzig', '109338');
INSERT INTO pleiadesmatch VALUES ('Rheinzabern', '109362');
INSERT INTO pleiadesmatch VALUES ('Lehen', '109404');
INSERT INTO pleiadesmatch VALUES ('Haute-Yutz', '109468');
INSERT INTO pleiadesmatch VALUES ('Aislingen', '118549');
INSERT INTO pleiadesmatch VALUES ('Altenstadt', '118559');
```

```
INSERT INTO pleiadesmatch VALUES ('Rottweil', '118572');
INSERT INTO pleiadesmatch VALUES ('Unterboebingen', '118601');
INSERT INTO pleiadesmatch VALUES ('Burladingen', '118618');
INSERT INTO pleiadesmatch VALUES ('Pfoerring', '118628');
INSERT INTO pleiadesmatch VALUES ('Echzell', '118657');
INSERT INTO pleiadesmatch VALUES ('Friedberg', '118681');
INSERT INTO pleiadesmatch VALUES ('Geislingen', '118690');
INSERT INTO pleiadesmatch VALUES ('Koesching', '118697');
INSERT INTO pleiadesmatch VALUES ('Koengen', '118706');
INSERT INTO pleiadesmatch VALUES ('Guenzburg', '118716');
INSERT INTO pleiadesmatch VALUES ('Okarben', '118777');
INSERT INTO pleiadesmatch VALUES ('Ladenburg', '118813');
INSERT INTO pleiadesmatch VALUES ('Munningen', '118815');
INSERT INTO pleiadesmatch VALUES ('Oberflorstadt', '118887');
INSERT INTO pleiadesmatch VALUES ('Kuenzing', '118919');
INSERT INTO pleiadesmatch VALUES ('Risstissen', '118933');
INSERT INTO pleiadesmatch VALUES ('Rueckingen', '118937');
INSERT INTO pleiadesmatch VALUES ('Schwabegg', '118957');
INSERT INTO pleiadesmatch VALUES ('Straubing', '118968');
INSERT INTO pleiadesmatch VALUES ('Stockstadt', '118976');
INSERT INTO pleiadesmatch VALUES ('Rottenburg', '118985');
INSERT INTO pleiadesmatch VALUES ('Burghoefe', '118986');
INSERT INTO pleiadesmatch VALUES ('Pfuenz', '119020');
INSERT INTO pleiadesmatch VALUES ('Waiblingen', '119041');
INSERT INTO pleiadesmatch VALUES ('Walheim', '119042');
INSERT INTO pleiadesmatch VALUES ('Vichy', '138186');
INSERT INTO pleiadesmatch VALUES ('Banassac', '138214');
INSERT INTO pleiadesmatch VALUES ('Cournon', '138294');
INSERT INTO pleiadesmatch VALUES ('Carrade', '138311');
INSERT INTO pleiadesmatch VALUES ('Espalion', '138344');
INSERT INTO pleiadesmatch VALUES ('Martres-de-Veyre', '138450');
INSERT INTO pleiadesmatch VALUES ('Mougon', '138456');
INSERT INTO pleiadesmatch VALUES ('Nouatre', '138491');
INSERT INTO pleiadesmatch VALUES ('Le Rozier', '138545');
INSERT INTO pleiadesmatch VALUES ('Toulon-sur-Allier', '138622');
INSERT INTO pleiadesmatch VALUES ('Aspiran', '148002');
INSERT INTO pleiadesmatch VALUES ('Lezoux', '167810');
```

```
INSERT INTO pleiadesmatch VALUES ('Baden', '177445');
INSERT INTO pleiadesmatch VALUES ('Luxeuil-les-Bains', '177568');
INSERT INTO pleiadesmatch VALUES ('Huefingen', '187326');
INSERT INTO pleiadesmatch VALUES ('Brigetio', '197180');
INSERT INTO pleiadesmatch VALUES ('Osijek', '197389');
INSERT INTO pleiadesmatch VALUES ('Narbonne', '246347');
INSERT INTO pleiadesmatch VALUES ('La Graufesenque', '246353');
INSERT INTO pleiadesmatch VALUES ('Bram', '246413');
INSERT INTO pleiadesmatch VALUES ('Crambade', '246505');
INSERT INTO pleiadesmatch VALUES ('Montans', '246505');
INSERT INTO pleiadesmatch VALUES ('Burgstall', '118850');
INSERT INTO pleiadesmatch VALUES ('Cannstatt', '118980');
INSERT INTO pleiadesmatch VALUES ('Heldenbergen', '118873');
INSERT INTO pleiadesmatch VALUES ('Holt', '79340');
INSERT INTO pleiadesmatch VALUES ('Inchtuthil', '89206');
INSERT INTO pleiadesmatch VALUES ('Newstead', '89304');
```

Anhang J: Quellcode der Apps

Der Quellcode der Apps befindet sich auf Grund der enormen Größe auf dem Datenträger.

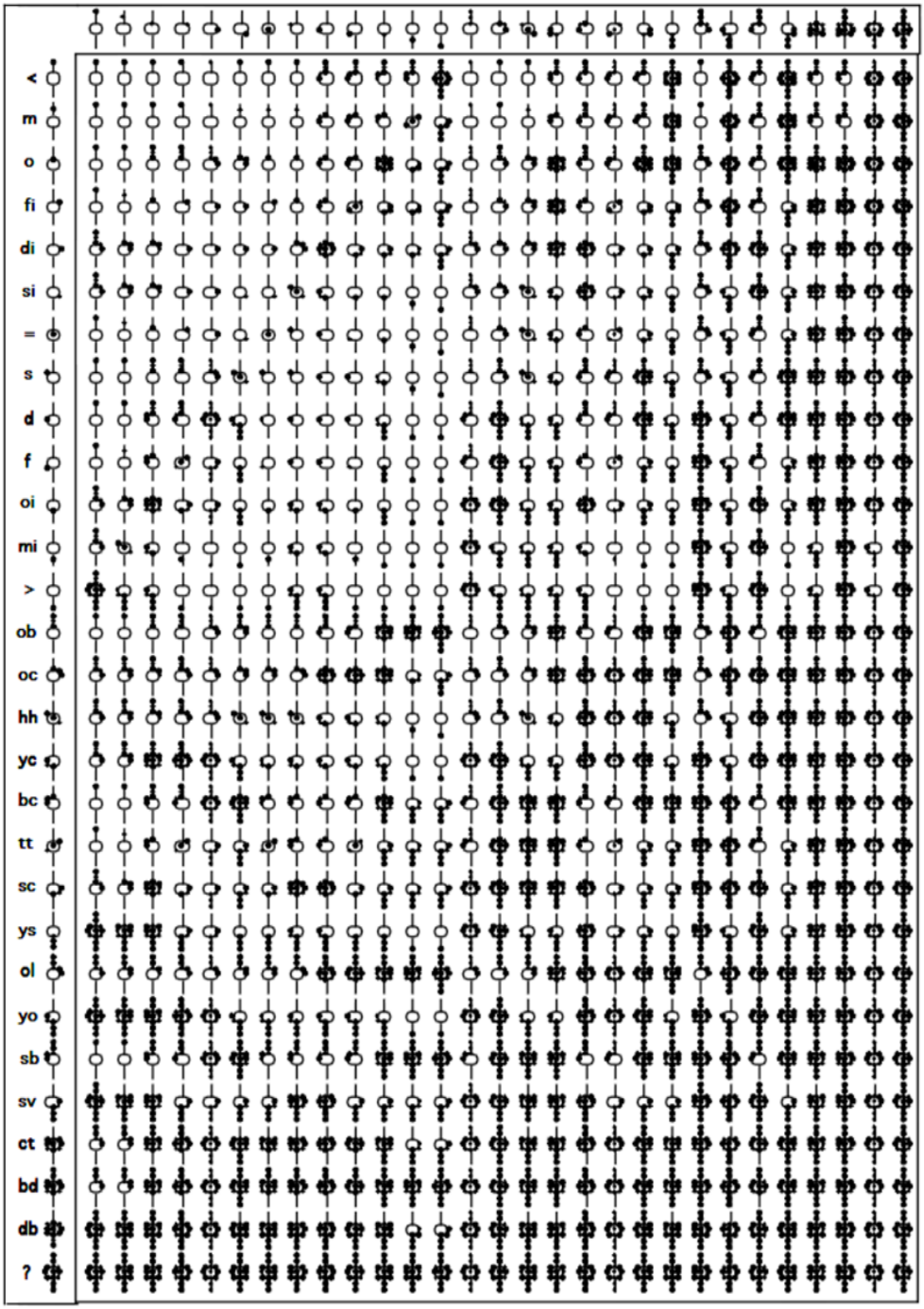
Anhang K: Allen und Freksa Schlussfolgerungen

B r2 C A r1 B	<	>	d	di	o	oi	m	mi	s	si	f	fi
"before" <	<	no info	< o m d s	<	<	< o m d s	<	< o m d s	<	<	< o m d s	<
"after" >	no info	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	> oi mi d f	>	>	>
"during" d	<	>	d	no info	< o m d s	> oi mi d f	<	>	d	> oi mi d f	d	< o m d s
"contains" di	< o m di fi	> oi di mi si	o oi dur con =	di	o di fi	oi di si	o di fi	oi di si	di fi o	di	di si oi	di
"overlaps" o	<	> oi di mi si	o d s	< o m di fi	< o m	o oi dur con =	<	oi di si	o	di fi o	d s o	< o m
"over-lapped-by" oi	< o m di fi	>	oi d f	> oi mi di si	o oi dur con =	> oi mi	o di fi	>	oi d f	oi > mi	oi	oi di si
"meets" m	<	> oi mi di si	o d s	<	<	o d s	<	f fi =	m	m	d s o	<
"met-by" mi	< o m di fi	>	oi d f	>	oi d f	>	s si =	>	d f oi	>	mi	mi
"starts" s	<	>	d	< o m di fi	< o m	oi d f	<	mi	s	s si =	d	< m o
"started by" si	< o m di fi	>	oi d f	di	o di fi	oi	o di fi	mi	s si =	si	oi	di
"finishes" f	<	>	d	> oi mi di si	o d s	> oi mi	m	>	d	> oi mi	f	f fi =
"finished-by" fi	<	> oi mi di si	o d s	di	o	oi di si	m	si oi di	o	di	f fi =	fi

Anhang 2: Allen Schlussfolgerungen [ALLEN, 1983]

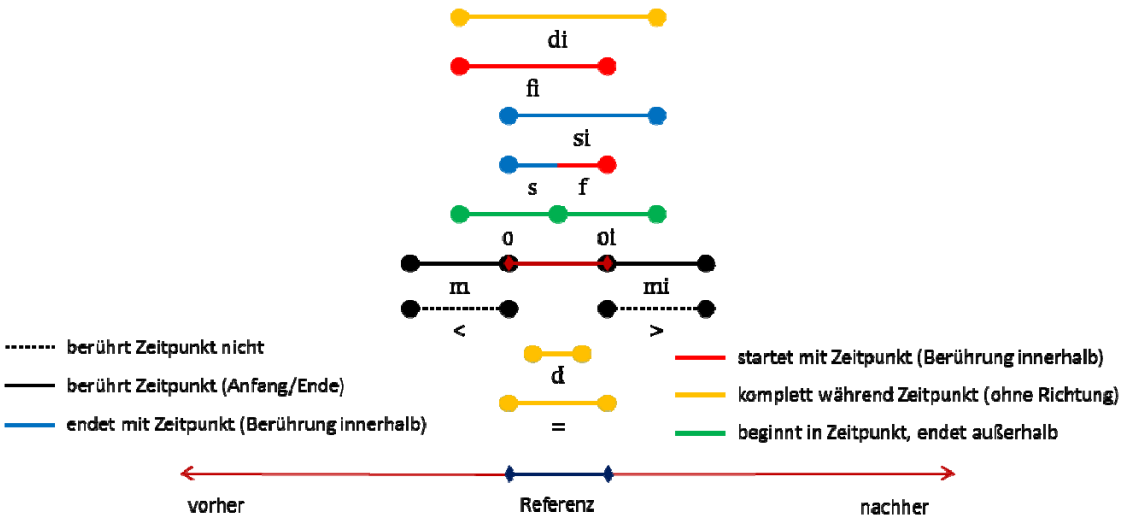
ICON	LABEL	MNEMONIC	ALLEN	CONSTRAINTS
	?		< m o fi di si = s d f oi mi >	none
	ol	*older*	< m o fi di	$\alpha < A$
	hh	*head to head with*	si = s	$\alpha = A$
	yo	*younger*	d f oi mi >	$\alpha > A$
	sb	*survived by*	< m o s d	$\omega < \Omega$
	tt	*tail to tail with*	fi = f	$\omega = \Omega$
	sv	*survives*	di si oi mi >	$\omega > \Omega$
	pr	*precedes*	< m	$\omega \leq A$
	bd	*born before death of*	< m o fi di si = s d f oi	$\alpha < \Omega$
	ct	*contemporary of*	o fi di si = s d f oi	$\alpha < \Omega, \omega > A$
	db	*died after birth of*	o fi di si = s d f oi mi >	$\omega > A$
	sd	*succeeds*	mi >	$\alpha \geq \Omega$
	ob	*older & survived by*	< m o	$\alpha < A, \omega < \Omega$
	oc	*older contemporary of*	o fi di	$\alpha < A, \omega > A$
	sc	*surviving contemporary of*	di si oi	$\alpha < \Omega, \omega > \Omega$
	bc	*survived by contemporary of*	o s d	$\omega > A, \omega < \Omega$
	yc	*younger contemporary of*	d f oi	$\alpha > A, \alpha < \Omega$
	ys	*younger & survives*	oi mi >	$\alpha > A, \omega > \Omega$

Anhang 3: Konzeptionelle Nachbarschaften [FREKSA, 1992]

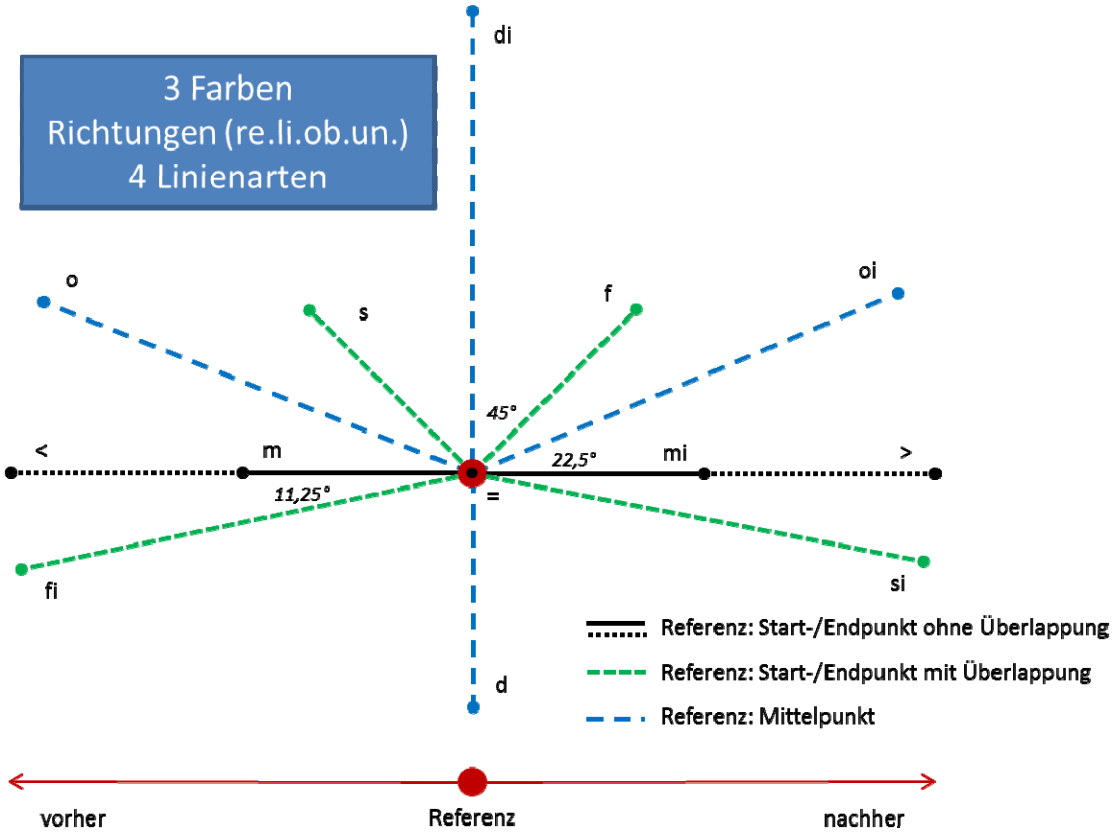


Anhang 4: Freksa Schlussfolgerungen [FREKSA, 1992]

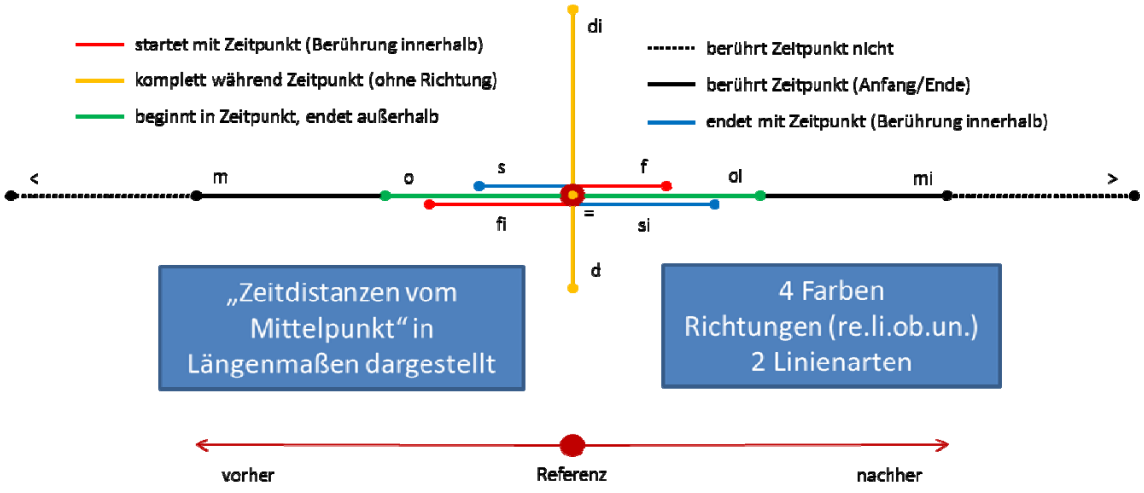
Anhang L: Visualisierungsideen



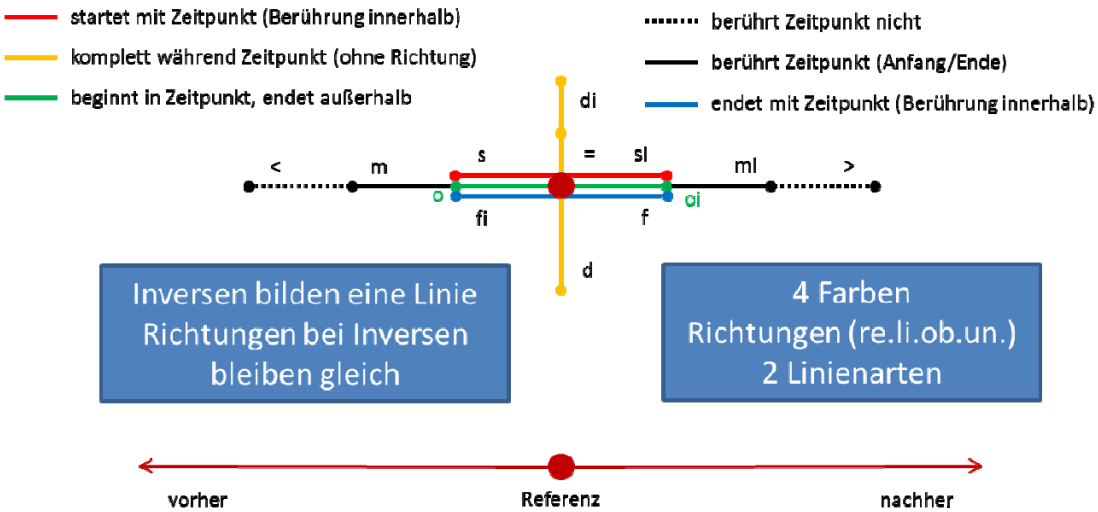
Anhang 5: Allen Relationen (Zeitstrahl)



Anhang 6: Allen Relationen (Graph 1)



Anhang 7: Allen Relationen (Graph 2)



Anhang 8: Allen Relationen (Graph 3)

Anhang M: Sonstiges

Weitere Dateien und Literatur sind Anhang M auf dem Datenträger zu entnehmen.

Quellenverzeichnis

Literatur

ALLEN, J. (1983): *Maintaining knowledge about temporal intervals*. Communications of the ACM, 26, 832–843.

BERNERS-LEE, T. (2009): *Linked Data*. 27. Juli 2006. <http://www.w3.org/DesignIssues/LinkedData.html>

BIZER ET AL. (2007): *How to Publish Linked Data on the Web*. 27. Juli 2007. <http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/20070727/>

DOERR ET AL. (2013): *CRMgeo: Linking the CIDOC CRM to GeoSPARQL through a Spatiotemporal Refinement* .TECHNICAL REPORT: ICS-FORTH/TR-435, April 2013. <http://cidoc–crm.org/docs/Technical%20Report435-CRMgeo.pdf>

EGENHOFER, M. (1993): *A Critical Comparison of the 4- Intersection and 9-Intersection Models for Spatial Relations: Formal Analysis*. Autocarto 11, R. McMaster and M. Armstrong (eds.),Minneapolis, MN, October 1993.

FELICETTI ET AL. (2007): *Open Source and Open Standards for using integrated geographic data on the web*. The 8th International Symposium on Virtual Reality, Archaeology and Cultural HeritageVAST (2007)D. Arnold, F. Niccolucci, A. Chalmers (Editors) <http://www.academia.edu/1566398/Open_Source_and_Open_Standards_for_Using_Integrated_Geographic_Data_on_the_Web>

FIELDING, R (2000): *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine (Dissertation), 2000.

FREKSA, C. (1991): *Conceptual Neighborhood and its role in temporal and spatial reasoning*. In: Singh, M., Travé-Massuyès, L. (eds.) Proc. of the IMACS Workshop on Decision Support Systems and Qualitative Reasoning, 181-187, North-Holland, Amsterdam 1991. <http://cindy.informatik.uni-bremen.de/cosy/staff/freksa/publications/ConceptualNeighborhood91.pdf>

FREKSA, C. (1992): *Temporal reasoning based on semi-intervals*. Artificial Intelligence 54, 199-227. <http://cindy.informatik.uni-bremen.de/cosy/staff/freksa/publications/TemReBaSeIn92.pdf>

GROSSNER, K. (2013A): *A space-time datatype for historical place?* Spatio-Temporal Technologies for the Arts and Humanities. DH 2013 Panel Discussion. Lincoln, Nebraska. 19 Juli 2013. <https://cga-

down-
load.hmdc.harvard.edu/publish_web/SpaceTime/DH2013_Grossner_SpaceTimeDatatype.pdf>

HAFT, M. (2013): *RDF als Verknüpfungsmethode zwischen geisteswissenschaftlichen Forschungsdaten und Geometrien am Beispiel des Projektes „Inschriften im Bezugssystem des Raumes“*. Bachelorarbeit im Fachbereich Informatik der Johannes Gutenberg Universität Mainz (nicht veröffentlicht) 2013.

HART, G. ET AL. (2013): *Linked Data: A Geographic Perspektive*. CRC/Taylor & Francis, Boca Raton 2013.

HEATH ET AL. (2011): *Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology*, 1:1, 1-136. Morgan & Claypool. <http://linkeddatabook.com/editions/1.0>

HENNEBRÜDER, S. (2007): *Hibernate*. Das Praxisbuch der Entwickler. Galileo Press Bonn 2007.

HITZLER, P. ET AL. (2008): *Semantic Web: Grundlagen* (eXamen.press). Springer, Berlin [u.a.] 2008.

HOLMEN & ORE (2009): *Deducing Event Chronology in a Cultural Heritage*. In: Bernard Frischer (Ed.): Making History Interactive: Computer Applications and Quantitative Methods in Archaeology (CAA); Proceedings of the 37th International Conference, Williamsburg, Virginia, United States of America, March 22 26, 2009. Vol. 2079. Archaeopress, Oxford. <http://www.edd.uio.no/artiklar/arkeologi/holmen_ore_caa2009.pdf>

ISAKSEN, L. (2011): *Archaeology and the Semantic Web*. PhD, University of Southampton, 2011. <http://eprints.soton.ac.uk/206421/>

KAUPPINEN ET AL. (2010): *Determining relevance of imprecise temporal intervals for cultural heritage information retrieval*. International Journal of Human-Computer Studies 68 (2010) 549–560. <http://kauppinen.net/tomi/temporal-relevance-ijhcs2010.pdf>

KOHR ET AL. (2012): Distrubuted Geodatabases in Archaeological Joint Research Projects. Vortrag der CAA in Groningen (Niederlande), 1.12.2012

KORDUAN, P. ET AL. (2008): *Geoinformation im Internet*. Technologien zur Nutzung raumbezogener Informationen im WWW. Wichmannverlag 2008 Heidelberg [u.A] 2008.

MEES, A. (2011): *Die Verbreitung von Terra Sigillata aus den Manufakturen von Arezzo, Pisa, Lyon und La Graufesenque*. Die Transformation der italischen Sigillata-Herstellung in Gallien. Verlag des Römisch-Germanischen Zentralmuseums, 2011.

OBE, R. ET AL. (2011): *PostGIS in action*. Manning Publications Co., Greenwich, Conn 2011.

PAPAKOSTAS, I. (2010): *Datenbankentwicklung mit PostgreSQL 9*. Wissen, das sich auszahlt. TEIA - Internet Akademie und Lehrbuch Verlag, Berlin 2010.

RICHARDSON, L. (2007): *RESTful Webservices*. O’Reilly Media, Inc. Sabastopol 2007.

RÖDER, D. (2010): *JPA mit Hibernate*. Java Persistence API in der Praxis, Entwickler Press Frankfurt am Main 2010.

SHAW ET AL. (2009): *LODE: Linking Open Descriptions of Events*. ASWC '09 Proceedings of the 4th Asian Conference on The Semantic Web. Seiten 153-167. <http://escholarship.org/uc/item/4pd6b5mh>

ULLENBOOM, C. (2012): *Java ist auch eine Insel*. 10. aktualisierte Auflage. Galileo Computing, Bonn 2012. <http://openbook.galileocomputing.de/javainsel/>

Websites und Online-Ressourcen

2PARTSMAGIC (2013): *REST-ful URI design*. <http://blog.2partsmagic.com/restful-uri-design> (abgerufen 04.12.2013)

ARACHNE (2013): *About Arachne Interfaces*. <http://arachne.uni-koeln.de/drupal/?q=de/node/234> (abgerufen 04.12.2013)

AYONAKAY (2012A): *WHAT'S THE DIFFERENCE BETWEEN WEB 1.0, WEB 2.0 AND WEB 3.0*. 12. September 2012. <http://alyonakay.wordpress.com/2012/09/23/whats-the-difference-between-web-1-0-web-2-0-and-web-3-0-2/> (abgerufen 04.12.2013)

BARKER (2011): *Pelagios Project Plan Part 1: Aims, Objectives and Final Output(s) of the project*. 21. Febraur 2011. <http://pelagios-project.blogspot.de/2011/02/pelagios-project-plan-part-1-aims.html> (abgerufen 04.12.2013)

BERNERS-LEE ET AL. (2001): *The Semantic Web: a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities*. In: Scientific American, 284 (5), S. 34–43, May 2001 <http://www.scientificamerican.com/article.cfm?id=the-semantic-web> (abgerufen 04.12.2013)

BERNERS LEE ET AL. (2005): *Uniform Resource Identifier (URI): Generic Syntax*. <http://tools.ietf.org/html/rfc3986> (abgerufen 04.12.2013)

BRICKLEY ET AL. (2010): *FOAF Vocabulary Specification 0.98*. Namespace Document 9 August 2010 - Marco Polo Edition. <http://xmlns.com/foaf/spec> (abgerufen 04.12.2013)

BUTLER ET AL.(2008): *The GeoJSON Format Specification*. <http://www.geojson.org/geojson-spec.html> (abgerufen 04.12.2013)

CENTOS (2013A): *CentOS*. <http://www.centos.org> (abgerufen 04.12.2013)

CENTOS (2013B): *CentOS Goals*. <http://www.centos.org/modules/tinycontent/index.php?id=5> (abgerufen 04.12.2013)

CIDOCCRMGMN (2013): *Hauptseite*. Deutsche Edition des CIDOC Conceptual Reference Model (Vers. 5.0.1). <http://cidoc-crm.gnm.de/wiki/Hauptseite> (abgerufen 04.12.2013)

DBPEDIA (2013A): *DBpedia Deutsch*. <http://de.dbpedia.org/> (abgerufen 04.12.2013)

DBPEDIA (2013B): *The DBpedia Ontology (3.9)*. <http://dbpedia.org/Ontology> (abgerufen 04.12.2013)

DCTERMS (2012): *DCMI Metadata Terms*. <http://dublincore.org/documents/dcmi-terms> (abgerufen 04.12.2013)

ENGLISHHERITAGE (2013): *MIDAS HERITAGE. THE UK HISTORIC ENVIRONMENT DATA STANDARD*. <http://www.english-heritage.org.uk/publications/midas-heritage/> (abgerufen 04.12.2013)

EPIDOC (2013): *EpiDoc: Epigraphic Documents in TEI XML*. <http://sourceforge.net/p/epidoc/wiki/Home> (abgerufen 04.12.2013)

FOAF (2013): *About FOAF*. <http://www.foaf-project.org/about> (abgerufen 04.12.2013)

GEOJSONWIKI (2013): *Users*. <http://wiki.geojson.org/Users> (abgerufen 04.12.2013)

GEONAMES (2013): *About GeoNames*. <http://www.geonames.org/about.html> (abgerufen 04.12.2013)

GEOVOCAB (2013): *GeoVocab*. <http://geovocab.org> (abgerufen 04.12.2013)

GESOSERVER (2013A): *Welcome*. <http://geoserver.org/display/GEOS/Welcome> (abgerufen 04.12.2013)

GESOSERVER (2013B): *What is GeoServer*. <http://geoserver.org/display/GEOS/What+is+GeoServer> (abgerufen 04.12.2013)

GETTY (2013A): *Getty Vocabularies*. <http://www.getty.edu/research/tools/vocabularies> (abgerufen 04.12.2013)

GETTY (2013B): *Vocabularies as LOD*. <http://www.getty.edu/research/tools/vocabularies/lod/index.html> (abgerufen 04.12.2013)

GILLIES, S. (2011): *Technical Introduction to Places*. 11. Februar 2013. <http://pleiades.stoa.org/help/technical-intro-places> (abgerufen 04.12.2013)

GILLIES, S. (2013A): *Pleiades RDF Vocabulary*. 3. April 2013. <http://pleiades.stoa.org/places/vocab> (abgerufen 04.12.2013)

GILLIES, S. (2013B): *Data for download*. <http://pleiades.stoa.org/downloads> (abgerufen 04.12.2013)

GISWIKI (2013): *Web Map Service*. <http://www.giswiki.org/wiki/Web_Map_Service> (abgerufen 04.12.2013)

GITHUB (2012): *Using the Pelagios API. The Pelagios API*. <https://github.com/pelagios/pelagios-cookbook/wiki/Using-the-Pelagios-API> (abgerufen 04.12.2013)

GITHUB (2013): *Adding Dataset Metadata with VoID*. <https://github.com/pelagios/pelagios-cookbook/wiki/Adding-Dataset-Metadata-with-VoID> (abgerufen 04.12.2013)

GROSSNER, K. (2013B): *Topotime: Qualitative reasoning for historical time*. 29. Juli 2013. <http://kgeographer.com/wp/topotime/> (abgerufen 04.12.2013)

GUTHUB (2012B): *Expressing Place References with OAC*. <https://github.com/pelagios/pelagios-cookbook/wiki/Expressing-Place-References-with-OAC> (abgerufen 04.12.2013)

HELMICH, M. (2012): *RESTful Webservices (1): Was ist das überhaupt?* 13. März 2013. <https://blog.mittwald.de/webentwicklung/restful-webservices-1-was-ist-das-uberhaupt/> (abgerufen 04.12.2013)

HeritageData (2013a): *About Heritage Data.* <http://www.heritagedata.org/blog> (abgerufen 04.12.2013)

HERITAGEDATA (2013B): *Vocabulary Providers.* <http://www.heritagedata.org/blog/vocabulary-providers> (abgerufen 04.12.2013)

HERITAGEDATA (2013C): *Vocabularies.* <http://www.heritagedata.org/blog/vocabularies-provided> (abgerufen 04.12.2013)

HERITAGESTANDARDS (2013): *FISH Interoperability Toolkit.* Last updated June 2011. <http://www.heritage-standards.org.uk/> (abgerufen 04.12.2013)

HERITAGESTANDARDS (2013B): *Documentation for MIDAS Spatial Schema Documentation.* <http://www.heritage-standards.org.uk/midas/docs/spatial/index.html?url=/midas/docs/spatial/spatial.html> (abgerufen 04.12.2013)

HIBERNATESPATIAL (2013): *Hibernate Spatial.* <http://www.hibernatespatial.org> (abgerufen 04.12.2013)

IANELLA ET AL. (2013): *vCard Ontology. For describing People and Organisations.* W3C Working Draft 24 September 2013. <http://www.w3.org/TR/vcard-rdf/> (abgerufen 04.12.2013)

ISAAC ET AL. (2009): *SKOS Simple Knowledge Organization System Primer.* W3C Working Group Note 18 August 2009. <http://www.w3.org/TR/skos-primer> (abgerufen 04.12.2013)

ITWISSEN (2013): *Web 3.0.* <http://www.itwissen.info/definition/lexikon/Web-3-0-web-3-0.html> (abgerufen 04.12.2013)

JANSEN (2012): *Thoughts on RESTful API Design.* Lessons learnt from designing the Red Hat Enterprise Virtualization API. 15. November 2012. <http://restful-api-design.readthedocs.org/en/latest/> (abgerufen 04.12.2013)

JAVA (2013): *Was ist die Java-Technologie und wozu wird sie benötigt?* <http://www.java.com/de/download/faq/whatis_java.xml> (abgerufen 04.12.2013)

JERSEY (2013): *Jersey - RESTful Web Services in Java.* <https://jersey.java.net> (abgerufen 04.12.2013)

JSON (2013): *Introducing JSON.* <http://www.json.org> (abgerufen 04.12.2013)

KANZAKI (2012): *Who's who description vocabulary.* 16. September 2012. <http://www.kanzaki.com/ns/whois#> (abgerufen 04.12.2013)

KLEIJN (2009): *Oracle kauft Sun: Was wird aus Java, MySQL und Openoffice?* 20. April 2009. <http://www.heise.de/open/artikel/Oracle-kauft-Sun-Was-wird-aus-Java-MySQL-und-OpenOffice-221803.html> (abgerufen 04.12.2013)

LEAFLET (2013A): *Leaflet*. <http://leafletjs.com> (abgerufen 04.12.2013)

LEAFLET (2013B): *Using GeoJSON with Leaflet*. <http://leafletjs.com/examples/geojson.html> (abgerufen 04.12.2013)

LEE, E. (2013): *Data Validator Vocabularies, with appropriate identifiers*. Last Update: 2. September 2004. <http://www.jiscmail.ac.uk/files/FISH/INSCRIPTION_IdentifiersV2.rtf> (abgerufen 04.12.2013)

LODCLOUD (2013A): *The Linking Open Data cloud diagram*. Letztes Update: 19. September 2011. <http://lod-cloud.net> (abgerufen 04.12.2013)

LOV (2013A): *Linked Open Vocabularies (LOV)*. <http://lov.okfn.org/dataset/lov> (abgerufen 03.12.2013)

LOV (2013B): *Linked Open Vocabularies (LOV)*. GML - OGC Geometry. <http://lov.okfn.org/dataset/lov/details/vocabulary_gml.html> (abgerufen 04.12.2013)

MADS (2013): *MADS Schema & Documentation*. 11 Juli 2013. <http://www.loc.gov/standards/mads> (abgerufen 04.12.2013)

MAVEN (2013): *Welcome to Apache Maven*. <http://maven.apache.org> (abgerufen 04.12.2013)

OAC (2013A): *About Open Annotation*. <http://www.openannotation.org/about.html> (abgerufen 04.12.2013)

OAC (2013B): *Open Annotation Data Model*. Community Draft, 08 February 2013. <http://www.openannotation.org/spec/core/> (abgerufen 04.12.2013)

OAC (2013D): *Open Annotation Data Model: Open Annotation Core*. Community Draft, 08 February 2013. <http://www.openannotation.org/spec/core/core.html> (abgerufen 04.12.2013)

OGC (2013A): *About OGC*. <http://www.opengeospatial.org/ogc> (abgerufen 04.12.2013)

OGC (2013B): *OGC Standards and Supporting Documents*. <http://www.opengeospatial.org/standards> (abgerufen 04.12.2013)

OPENJDK (2010): *OpenJDK FAQ*. Bearbeitungsdatumg: 18.10.2010. <http://openjdk.java.net/faq/> (abgerufen 04.12.2013)

OPENLAYERS (2013): *OpenLayers*. <http://openlayers.org> (abgerufen 04.12.2013)

OPENRDF (2013A): *Sesame 2.7 user documentation*. <http://openrdf.callimachus.net/sesame/2.7/docs/users.docbook?view> (abgerufen 04.12.2013)

OPENRDF (2013B): *About*. <http://www.openrdf.org/about.jsp> (abgerufen 04.12.2013)

OPENRDF (2013C): *Consulting*. <http://www.openrdf.org/consulting.jsp> (abgerufen 04.12.2013)

ORTIZ (2012): *45 Ways to Communicate Two Quantities*. <http://blog.visual.ly/45-ways-to-communicate-two-quantities/> (abgerufen 04.12.2013)

OSGEO (2013): *Geography Markup Language (GML)*. <http://live.osgeo.org/de/standards/gml_overview.html> (abgerufen 04.12.2013)

PELAGIOS (2013): *About*. <http://pelagios-project.blogspot.de/p/about-pelagios.html> (abgerufen 04.12.2013)

PERRY ET AL. (2012): *OGC GeoSPARQL - A Geographic Query Language for RDF Data*. Bearbeitungsdatum: 10.09.2012. <http://www.opengis.net/doc/IS/geosparql/1.0> abgerufen 04.12.2013)

PLEIADES (2013A): *Pleiades*. A community-built gazetteer and graph of ancient places . <http://pleiades.stoa.org> (abgerufen 04.12.2013)

PLEIADES (2013C): *About Pleiades*. <http://pleiades.stoa.org/home> (abgerufen 04.12.2013)

POSTGIS (2013): *POSTGIS: PostGIS 2.1.2dev Manual*. SVN Revision (12147). <http://postgis.net/docs/manual-2.1/> (abgerufen 04.12.2013)

RDFA (2013): *Linked Data in HTML*. <http://rdfa.info> (abgerufen 04.12.2013)

SPORNY ET AL. (2012): *RDFa Lite 1.1*. W3C Recommendation 07 June 2012. <http://www.w3.org/TR/rdfa-lite/> (abgerufen 04.12.2013)

TATE (2013): *Collection data*. <http://www.tate.org.uk/about/our-work/digital/collection-data> (abgerufen 04.12.2013)

TEI (2013): *TEI: Text Encoding Initiative*. <http://www.tei-c.org> (abgerufen 04.12.2013)

VOGEL, L. (2012): *JAXB Tutorial*. Bearbeitungsstand: 23.11.2012. <http://www.vogella.com/articles/JAXB/article.html> (abgerufen 04.12.2013)

W3C (2013): *Über das World Wide Web Consortium (W3C)*. <http://www.w3c.at/about/overview.html> (abgerufen 04.12.2013)

W3TECHS(2013): *Usage statistics and market share of Linux for websites*. <http://w3techs.com/technologies/details/os-linux/all/all> (abgerufen 22.10.2013)

WIKIPEDIA (2013A): *European Petroleum Survey Group Geodesy*. Bearbeitungsstand: 21. Juli 2013, 19:17 UTC. <http://de.wikipedia.org/wiki/European_Petroleum_Survey_Group_Geodesy> (abgerufen 04.12.2013)

WIKIPEDIA (2013B): *Dublin Core*. Bearbeitungsstand: 24. Juli 2013, 12:58 UTC. <http://de.wikipedia.org/w/index.php?title=Dublin_Core&oldid=120844919> (abgerufen 04.12.2013)

ZAWALSKI, A. (2013): *JAVA-Grundlagen und ihre Vorteile*. <http://www.ahhhjah.de/java-grundlagen-und-ihre-vorteile> (abgerufen 04.12.2013)

Abbildungen

5STARDATA (2013): http://5stardata.info/5star-steps.png

AYONAKAY (2012B): http://alyonakay.files.wordpress.com/2012/09/websummary2.jpg?w=510

FERGI (2013): http://www.fergi.uni-osnabrueck.de/module/geodstd/inhalt/8/bilder/9imoperationen_screen.gif

HS (2013A): http://www.heritage-standards.org.uk/midas/docs/1-1/actor/midas_actor_p1.png

HS (2013B): http://www.heritage-standards.org.uk/midas/docs/1-1/spatial/midas_spatial_p1.png

KEY (2013): http://keybridgeglobal.com/keybridge/javax.faces.resource/technology/interop/sfa-geometrytypes.png.xhtml;jsessionid=39a20b3f0e7f72ced071c45b7318?ln=images

KGEO (2013A): http://kgeographer.com/wp/wp-content/uploads/2013/07/holmen_1.png

KGEO (2013B): http://kgeographer.com/wp/wp-content/uploads/2013/07/kaupinnen_1.png

KINDREDBRITAIN (2013): http://kindred.stanford.edu

LODcloud (2013b): http://lod-cloud.net/versions/2011-09-19/lod-cloud_colored.png

OAC (2013C): http://www.openannotation.org/spec/core/images/intro_model.png

OAC (2013E): http://www.openannotation.org/spec/core/images/anno_model1.png

OAC (2013F): http://www.openannotation.org/spec/core/images/motivations.png

OGC (2013C): http://www.opengeospatial.org/standards/sfa

OSW (2013): http://open.semantic-web.at/display/OGDW/6.3+Open-Data-5-Stern-Modell+von+Tim+Berners-Lee

PLEIADES (2013B): http://pleiades.stoa.org/images/entities.png/image_large

PUNDIT (2013A): http://youtu.be/RzBm8cb-a4U

PUNDIT (2013B): http://www.thepund.it/wp-content/uploads/2013/03/annotation_small.png

RESTFUL (2013): http://restful-api-design.readthedocs.org/en/latest/_images/concepts.png

RGZM (2013): http://www.rgzm.de/Samian/Queries/Cat29FullOutput.cfm?SerialNumber=0002465&Potter=Secundus%20ii&DieNo=31b

WIKIMEDIA (2006): http://commons.wikimedia.org/wiki/File:Servlet.png