Entwurfsbeschreibung Gesamtprojekt

Thema: SemanticChess
Gruppe: swp13-sc

Verantwortlich: L.Kohlmeyer

Version 2
Letzte Änderung: 22.04.13

Inhaltsverzeichnis

1 Allgemeines

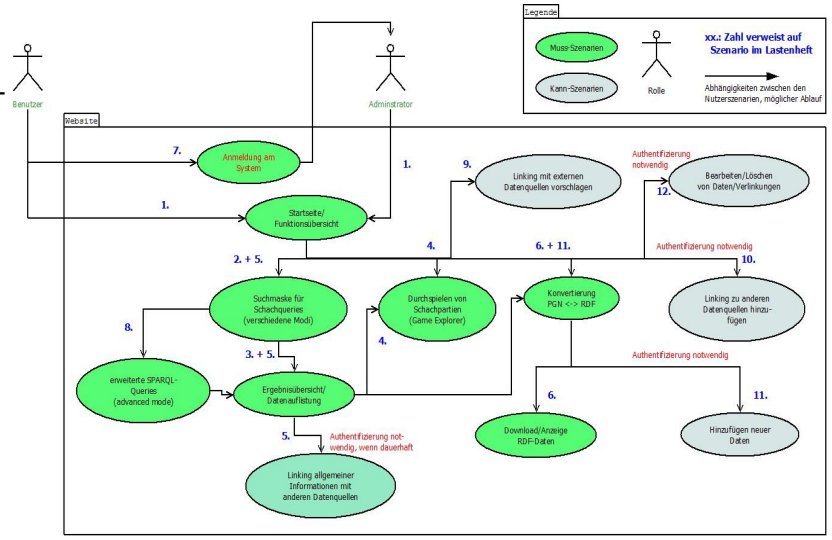
Schach beschäftigt die Menschen mindestens seit dem 6. Jahrhundert. Die Erfindung des modernen Computers eröffnete Schachinteressierten aus aller Welt eine Fülle an neuen Möglichkeiten. Herkömmliche Schachcomputer oder Schachprogramme nutzen vor allem die hohe Rechenleistung der Maschinen aus sowie die Tatsache, dass Schach ein rein deterministisches Spiel ist, um so mögliche zum Sieg führende Züge zu berechnen. Mit unserem Projekt verfolgen wir einen alternativen Ansatz zur Handhabung von Schachspielen mit dem Computer. In der langen Geschichte des Schachs wurde eine Fülle an Spielen aufgezeichnet. Diese Fülle an Spielen kann nun statistisch ausgewertet werden, um wiederum zu bestmöglichen Zügen in konkreten Schachsituationen zu gelangen. Ein jüngerer Trend in der Informatik ist das Konzept des Semantic Webs. Dieses zeichnet sich durch die Speicherung der Zusammenhänge von Information aus. Dadurch weiß es quasi um die Semantik der in ihm enthaltenen Informationen, was zielgerichtete Abfragen ermöglicht, wie zum Beispiel im weitesten Sinne Abfragen nach den besten Zügen, aber auch Fragen nach Informationen zu Spielerdaten.

Das Ziel dieser Arbeit ist aber bei weitem nicht die Entwicklung einer alternativen Schach-KI, es ist eher als das Legen des Fundamentes dafür zu verstehen.
Unsere Ergebnisse sollen als Web-Applikation letztlich für jeden zur freien Verfügung stehen. Ein Webserver dient dabei der Verwaltung der aufgezeichneten Spiele.

2 Produktübersicht

Das Softwareprodukt soll dazu dienen mithilfe einer OpenSource-Schachengine und einer eigens konzipierten Web-Oberfläche Schachpartien in einer Web-Applikation zu simulieren und durchspielen zu lassen. Dabei soll ein Datenbestand von rund 2 Millionen Schachpartien auf dem Webserver zur Auswertung verfügbar gemacht werden. Die Auswertung der Partien soll dabei durch Nutzen von Methoden des Semantic Webs erfolgen. Es soll auch möglich sein, eigene Partien hinzuzufügen und selbst die Konvertierungsfunktion zu nutzen.

Die im Vorprojekt abgeschlossene Softwarestudie sollte dabei über die Durchführbarkeit und das Zusammenspiel der wichtigsten Komponenten einer solchen Software Aufschluss geben. Darüber hinaus stellt der im Rahmen dieser Studie erstellte Prototyp die grundlegenden Softwareelemente bereit, die nun im Hauptprojekt zu einem finalen Produkt ergänzt und miteinander verbunden werden sollen.
Der Schwerpunkte der Funktionalität des Prototypen aus dem Vorprojekt sind auf die Erstellung der Schachontologie, die Konvertierung von PGN-Dateien in ein RDF-Daten-Format und auf eine Vorversion der Web-Oberfläche ausgerichtet. Im weiteren Projektverlauf sollen die nun im RDF-Format gespeicherten Daten mit Informationen aus der DBpedia durch Verlinkung angereichert werden, eine vereinfachter Suche soll entwickelt werden, eine Schach-Engine soll das Durchspielen bisheriger Spiele ermöglichen und alles gemeinsam soll den Webseiten-Prototypen mit Funktionalität füllen.
Konkreter gesagt hat der Nutzer aus dem Vorprojekt die Möglichkeit, PGN-Daten per copy-paste oder Fileupload dem Server zu übergeben, woraufhin die Daten vom Parser zur Weiterverarbeitung aufbereitet werden, um anschließend nach der Konvertierungs-Vorschrift in RDF-Daten umgewandelt zu werden.
Außerdem soll der Nutzer Partien mittels eines Game Explorers in der Schach-Engine durchspielen und Abfragen an die Spieldatenbank richten können. Die Abfrage soll dabei vereinfacht als auch mit SPARQL erfolgen können. Die gesamte Produktübersicht ist dem nebenstehenden



Use-Case-Diagramm zu entnehmen, das auch schon im Lastenheft Verwendung fand.

3 Struktur und Entwurfsprinzipien

Die Anwendung samt GUI soll im Web als Java Servlet spezifiziert und implementiert werden. Dies wird über den Open-Source-Webserver Apache Tomcat realisiert, welcher bereits in dem zur Verfügung stehenden Server integriert wurde. Im Allgemeinen besteht der Prototyp aus einer GUI, mit welcher der Client in seinem Browser interagieren kann, einer Schachontologie, welche Informationen zu verschiedenen Schachpartien im RDF bereit hält, einer von Virtuoso verwalteten Datenbank, einem Parser, der PGN-Dateien in konvertierbare Daten zerteilt sowie einem Konverter, welcher PGN-Daten in RDF-Form umwandelt. An den Prototypen sollten sich alle weiteren Implementierungsschritte nahtlos anschließen können. Die Ontologie soll nach Nutzung von Limes Spielerdaten der DBPedia bereitstellen, die Datenbestände in Virtuoso sollen abgefragt werden können, eine Schacheingine soll das Spielen bzw. Durchlaufen von Partien ermöglichen und alles zusammen soll auf der Webseite zur freien Verfügung stehen.

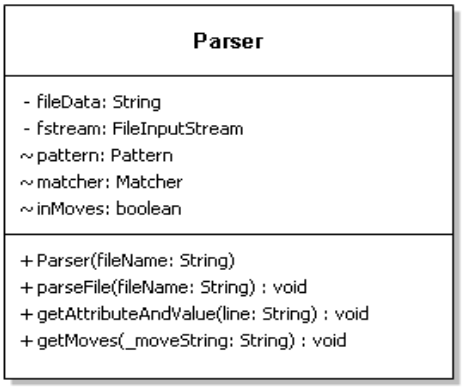
4 Struktur und Entwurfsprinzipien einzelner Pakete

Pakete werden hier wie im Lastenheft dargestellt. Muss-Aspekte sind durch ein M gekennzeichnet, Kann-Aspekte durch K. Grundsätzlich gilt hier: **Detailliertere Informationen, Ergänzungen oder Ausführungen folgen, sobald das Softwareprojekt einen entsprechenden Entwicklungsstand erreicht hat.**

1. Konvertierung PGN <-> RDF

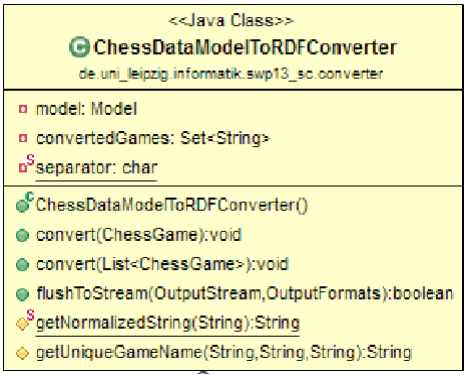
1.M.1 Konzipierung einer Schachontologie in RDF-Schema

Es wurden mehrere Ontologien erstellt. Die erste stellte sich als zu umfangreich und zu spezifisch heraus, so dass eine zweite einfacher gehaltene Ontologie erstellt wurde. Grundsätzlich fasst sie Schachspiele als eine Sequenz von Zügen auf, wobei Züge Symbole, Kommentare und Namen haben können und Spiele Ergebnisse und weitere relevante Daten besitzen können. Die Ontologie soll nun im weiteren Projektverlauf iterativ wachsen – und kann auch noch nach Abschluss des Projektes iterativ erweitert werden. Für die Erstellung wurde das freie Softwaretool Protégé benutzt. Dieses ermöglicht den Export der Ontologie in verschiedene im Web gebräuchliche Formate, wie z. B. Turtle, XML/RDF.



1.M.2 Erstellen einer Konvertierungsvorschrift zum Überführen von PGN-Daten in RDF-Daten

Zunächst wird die mitgegebene PGN-Datei dem Parser übergeben. Dieser stellt die PGN-Daten als Speicherdaten für den Computer bereit, so dass Letzterer damit arbeiten kann. Die geparsten Daten werden nun mit Hilfe von Jena konvertiert, so dass die Daten letztlich in einem RDF-Format vorliegen.



1.M.3 Speicherung konvertierter Daten in Triple Stores

Die konvertierten Daten werden auf dem Server in Triple Stores gespeichert, wo weiter mit ihnen gearbeitet werden kann, indem an das Datenverwaltungssystem Virtuoso Anfragen gerichtet werden. Die technischen Schwierigkeiten, die hier auftraten, kamen zu Stande aufgrund mehrerer Virtuoso-Versionen, die auf dem Server lagen. Das Löschen eine Konfiguratinsdatei behob diese Schwierigkeiten.

1.K.1 Eröffnungsontologie für die Anzeige von Eröffnungsnamen beim Browsen und suchen von Eröffnungen

Eine Eröffnungsontologie kann ähnlich wie die Schachontologie erstellt werden und sollte dann in das System integriert werden, in der Art, dass bezüglich dieser Ontologie auch konvertiert werden kann.

2. Linking mit anderen Datenquellen

2.M.1 Verlinkung mit der Dbpedia

Die Verlinkung mit DBPedia soll mittels Limes erfolgen.

2.K.1 Verlinkung mit anderen geeigneten semantischen Wissensbeständen
Es stünde zunächst die Suche nach anderen geeigneten Wissensbeständen im Vordergrund. Eine Anknüpfung könnte möglicherweise auch mit Limes erfolgen. Die Verlinkung steht aber nicht im Vordergrund des Projektes und wird daher nicht erfolgen.

3. Abfrage
3.M.1 Erleichterte Abfrage für Ungelernte
Hier gilt es ein intuitiv bedienbares Suchsystem zu entwickeln, das funktional umfangreich, mindestens jedoch einfacher als SPARQL-Code gehalten ist. Die einfacheren Anfragen werden intern in SPARQL-Code übersetzt. Dafür werden aus einer Suchmaske bestimmte Informationen zu an Spielen beteiligten Spielern oder Spielen selbst ausgegeben. Als Ergebnisraum ist es möglich zwischen Spielen und Spielern zu unterscheiden. Die Funktionalität der Suche soll in den folgenden Wochen noch in die Oberfläche integriert werden. Zur Generierung der SPARQL-Anfragen wurde auf diverse String-Konstanten zurückgegriffen, die nach dem Baukastenprinzip zusammen gebaut werden können und das Klassendiagramm etwas in die Länge ziehen.

3.M.2 Advanced mode mittels eigener SPARQL-Anfragen
Auch die SPARQL-Anfragen sollen direkt über die Benutzeroberfläche gestellt werden können. Virtuoso, das die Daten der Partien in dem Triple Store speichert, ist dann in der Lage, auf die SPARQL-Anfragen zu reagieren. Hierfür ist es nur notwendig entsprechende Schnittstellen zwischen der Oberfläche und Virtuoso zu schaffen. Die Oberfläche wird dann um ein Eingabefeld für die SARQL-Abfrage und ein Ausgabefeld ergänzt.

3.M.3 Statistische Auswertung für Gewinnwahrscheinlichkeiten mittels einer erstellten Spieldistanz
Die Spieldistanz soll angeben, wie sehr sich zwei Partien bis zu einem gewissen Punkt gleichen. Die Gewinnwahrscheinlichkeiten sollen sich nun aus einem statistischen Vergleich aller ähnlichen Spiele ergeben.

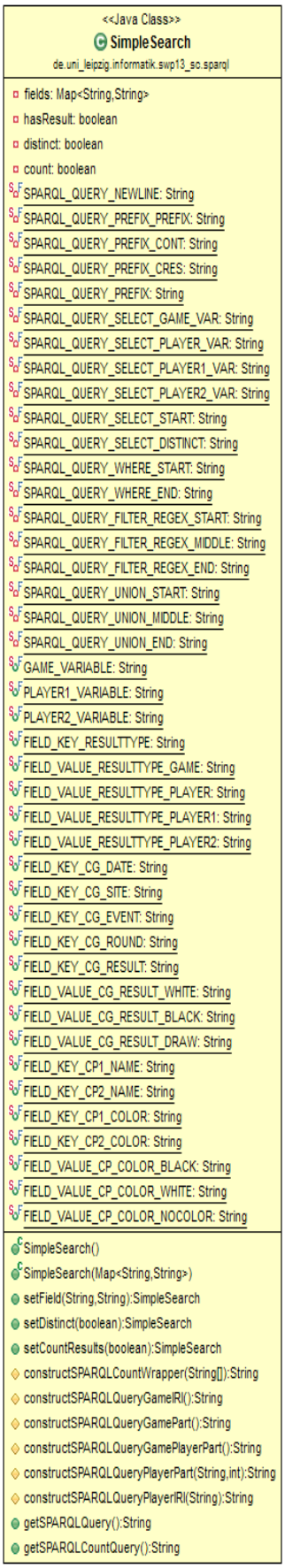
3.K.1 Abfragen für die Eröffnungsontologie
Ist erstmal eine Eröffnungsontologie vorhanden, so ist es auch sinnvoll vereinfachte Abfragen für diese möglich zu machen. Dies geschieht in Analogie zu den anderen Abfragen.

4. Schachengine
4.M.1 Nutzen einer Schachengine zum Spielen von Partien

Vermutlich wird Carballo genutzt werden, d.h., auf der Webseite integriert werden, um Partien (durch) zu spielen. Dies soll zunächst ohne Ausnutzung der semantischen Wissensbasis geschehen.

4.K.1 Verknüpfung der Schachengine mit der semantischen Wissensbasis
Um die Engine mit der semantischen Wissens basis zu verknüpfen, ist es wichtig die Schnittstellen, zwischen beiden System auszumachen. Die Engine sollte in der Lage sein die Gewinnwahrscheinlichkeiten abzufragen, die auf der Web-Oberfläche zur Verfügung gestellt werden.

4.K.2 Spiel zwischen Engine ohne Wissensbasis gegen Engine mit Wissensbasis
Bei der Umsetzung dieses Ziels, kommt es vor allen auf die KIs der Engines an.

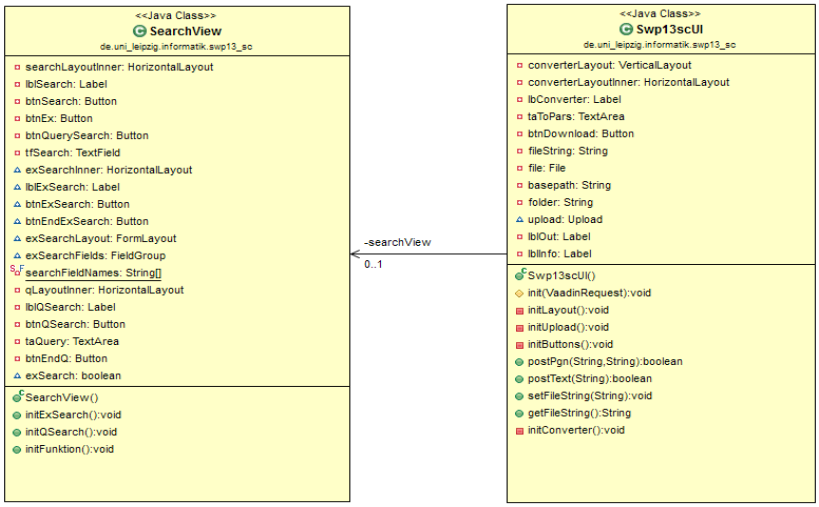


Diese müssen so zusammengebracht werden, dass sie für ein Publikum sichtbar auf der Oberfläche der Engine gegeneinander spielen. Das Spiel sollte aufgezeichnet werden. Es sollte darüber aufschluss geben, ob eine KI mit semantischem Wissen gegenüber einer ohne diesem Wissen Vorteile hat.

4.K.3 Einrichtung einer PVP-Funktion
Eine Spieler-gegen-Spieler-Funktion erfordert die Anmeldung am System, falls die Engine diese Funktionalität nicht von sich aus bietet. Dieses Ziel wird nicht umgesetzt werde, da es nicht in so engem Zusammenhang zu den anderen Zielen steht und auch nur auf Wunsch der Projektgruppe hier aufgenommen wurde.

5. Weboberfläche

Die Web-Oberfläche soll so übersichtlich und einfach bzw. komfortabel wie möglich gestaltet werden.
Um dies zu erreichen soll durch Vaadin eine in Java geschriebene WebGUI aufgesetzt und mithilfe des Tomcatservers visualisiert werden. Die GUI umfasst die folgenden Aspekte:



5.M.1 Suchmaske für vereinfachte Abfragen sowie SPARQL-Abfragen
Die SPARQL-Abfragen können direkt in einem Textfeld auf der GUI eingegeben werden. Für die vereinfachte Suche, soll in eine Suchmaske mit mehreren Feldern nach Verschiedenen Attributen gesucht werden. Ergebnisraum soll aus Spielen oder Spielern bestehen.

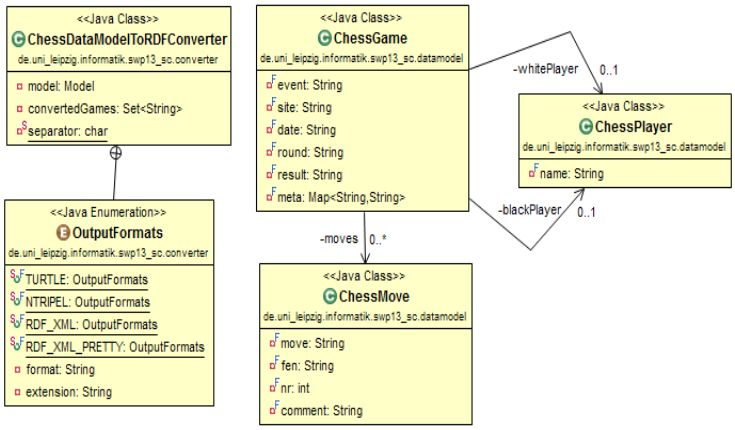
5.M.2 Grafisches Schachbrett mit Figuren, Integration einer Schachengine, Durchspielen von Partien
Eine Engine soll auf der Oberfläche integriert werden und optisch den Kern darstellen.

5.M.3 Anzeige der Gewinnwahrscheinlichkeiten der nächsten Züge
Hierfür genügen simple Balkendiagramme, die die Wahrscheinlichkeiten für Sieg/unentschieden/Niederlage angeben.

5.M.4 Konvertierungs-/Uploadfunktion
Ein Auswahlfeld, sowie ein Bestätigungsbutton sollen die Konvertierungs-/Uploadfunktion darstellen.

5.K.1 Grafische Integration der Kann-Ziele anderer Arbeitspakete
Dies bedeutet vor allem die Erweiterung der Suchmaske um die Eröffnungsontologie.

5 Datenmodell


<<Java Class>>
ChessDataModelToRDFConverter
de.uni_leipzig.informatik.swp13_sc.converter
model: Model
convertedGames: Set<String>
separator: char
<<Java Enumeration>>
OutputFormats
de.uni_leipzig.informatik.swp13_sc.converter
TURTLE: OutputFormats
NTRIPEL: OutputFormats
RDF_XML: OutputFormats
RDF_XML_PRETTY: OutputFormats
format: String
extension: String
<<Java Class>>
ChessGame
de.uni_leipzig.informatik.swp13_sc.datamodel
event: String
site: String
date: String
round: String
result: String
meta: Map<String,String>
-whitePlayer
0..1
-blackPlayer
0..1
-moves
0..*
<<Java Class>>
ChessPlayer
de.uni_leipzig.informatik.swp13_sc.datamodel
name: String
<<Java Class>>
ChessMove
de.uni_leipzig.informatik.swp13_sc.datamodel
move: String
fen: String
nr: int
comment: String

6 Testkonzept

Die automatischen Tests werden während der Implementierung in der Entwicklungsumgebung angezeigt und stellen weniger Probleme dar, da sie vom Programmierer nahezu akut gelöst werden können. Den manuellen Tests muss hingegen größere Aufmerksamkeit geschenkt werden. Während der Programmierung sollten einzelne Komponenten auf Funktionstüchtigkeit getestet werden. Z.B. sollte jede Klasse eine eigene „Testmain“ mit Ein- und Ausgabe enthalten, außerdem sollte dabei auf Grenzfälle wie verschiedene Datentypen, unvorhergesehene Eingaben, geachtet werden. Beim Testen muss auf die syntaktische und semantische Funktionalität geachtet werden. Wenn alles getestet und zusammengefügt wurde, dann gilt es, die Gesamtheit auf Funktionstüchtigkeit zu testen. Hierbei ist es von großer Bedeutung, dass viele Leute testen können, die den Quellcode nicht einmal gesehen haben. Somit hat man die Möglichkeit, mit vielen Testern schnell eine Liste von Bugs zu erstellen, die von den Programmieren dann beseitigt werden sollen. In unserem Beispiel können in den Zweierteams von Programmierern die automatischen Tests voraussichtlich gut in der Entwicklungsumgebung gehandhabt werden. Die gesamtheitlichen Test werden andererseits sehr viel Zeit in Anspruch nehmen. Folgende Fragen müssen im Zusammenspiel der Systemkomponenten analysiert werden:

Wurde die Ontologie auch genau so konvertiert wie geplant?
Ist das Nutzerinterface fehlerfrei?
Ist das Nutzerinterface benutzerfreundlich?
Funktionieren alle Schnittstellen zwischen Server, Datenbank und Client wie geplant?

Mit mehreren Testdurchläufen und Updates wollen wir versuchen alle Fragen mit einem signifikantem „Ja“ zu beantworten.

7 Glossar

Ontologie
Explizite formale, maschinenlesbare Spezifikationen einer Konzeptualisierung (Beziehungen, Ableitungsregeln) zu einem bestimmten Wissensbereich bezeichnet man als Ontologien. Diese ermöglichen es ein Netzwerk von Informationen mit logischen Relationen darzustellen, innerhalb dem logische Schlussfolgerungen sowie Gewährleistung der Gültigkeit möglich sind. Ontologien beschreiben RDF-Daten formal für andere Systeme und ermöglichen eine Validierung.

Parser
Ist ein bestimmtes Programm zur Zerlegung oder Umwandlung einer beliebigen Eingabe in ein für die Weiterverarbeitung brauchbares Format.

SPARQL
(**S**PARQL **P**rotocol **A**nd **R**DF **Q**uery **L**anguage) ist eine graphbasierte Standard-Abfragesprache für RDF Formate.

Vaadin
Ist ein freies Webanwendungs-Framework für Internetapplikationen unter der Apache Lizenz 2.0. Die Entwicklung kann in purem Java erfolgen, die browserseitige Unterstützung wird vom Framework übernommen. Vaadin steht im Gegensatz zu Browserplugins oder Java-Script basierten Lösungen, da ein Großteil der Programmlogik serverseitig ausgeführt wird.

Virtuoso
Ist eine Datenbank-Engine, die Funktionalitäten wie z.B. RDBMS, RDF, XML, web-application-server, virtual-database in einem einzigen System kombiniert.