Entwurfsbeschreibung Gesamtprojekt

Thema: SemanticChess
Gruppe: swp13-sc

Verantwortlich: L.Kohlmeyer

Version 6
Letzte Änderung: 20.05.13

Inhaltsverzeichnis

1 Allgemeines

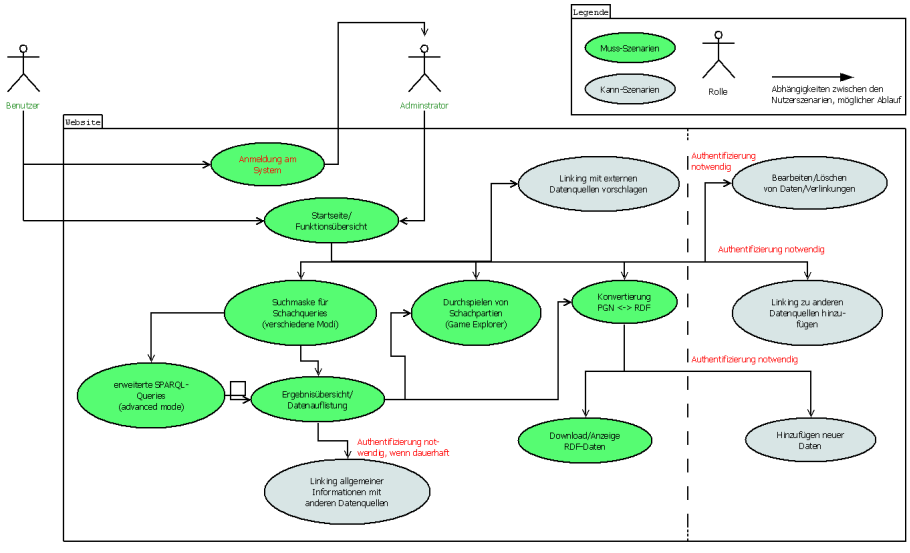
Schach beschäftigt die Menschen mindestens seit dem 6. Jahrhundert. Die Erfindung des modernen Computers eröffnete Schachinteressierten aus aller Welt eine Fülle an neuen Möglichkeiten. Herkömmliche Schachcomputer oder Schachprogramme nutzen vor allem die hohe Rechenleistung der Maschinen aus sowie die 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 wird ein Datenbestand von rund 2 Millionen Schachpartien auf dem Webserver zur Auswertung verfügbar gemacht, der aus PGN-Dateien konvertiert wurde. Die Auswertung der Partien erfolgt dabei durch Nutzen von Methoden des Semantic Webs. Es ist auch möglich 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 im Hauptprojekt zu einem finalen Produkt ergänzt und miteinander verbunden wurden.
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 wurden die nun im RDF-Format gespeicherten Daten mit Informationen aus der DBpedia durch Verlinkung angereichert, eine vereinfachte sowie eine komplexe Suche wurden entwickelt und eine Schach-Engine ermöglicht das Durchspielen bisheriger Spiele, sowie das Spielen neuer Spiele. Die Funktionalitäten sind auf der Webseite des Semantic Chess Projects abrufbar.



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 das Produkt aus einer GUI, mit welcher der Client in seinem Browser interagieren kann, einer Schachontologie, welche den Aufbau von Informationen zu verschiedenen Schachpartien im RDF beschreibt, einer von Virtuoso verwalteten Datenbank, einem Parser, der PGN-Dateien in konvertierbare Daten zerteilt sowie einem Konverter, welcher PGN-Daten in RDF-Form umwandelt. Darüber hinaus stellt das Produkt weitere Spielerdaten der DBPedia durch die Nutzung von LIMES bereit, die Datenbestände in Virtuoso sind abfragbar (einfach und komplex), eine Schachengine ermöglicht das Spielen bzw. Durchlaufen von Partien und die Funktionalitäten stehen auf der Webseite zur freien Verfügung bereit.

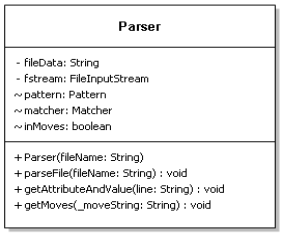
4 Struktur und Entwurfsprinzipien einzelner Pakete

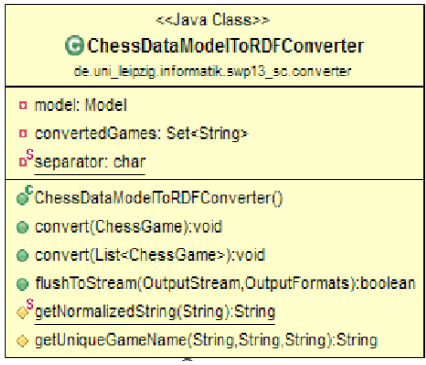
Pakete werden hier wie im Projektvertrag dargestellt. Muss-Aspekte sind durch ein M gekennzeichnet, Kann-Aspekte durch K.

*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 ist im weiteren Projektverlauf iterativ gewachsen, z.B um eine FEN-Berechnung 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 einer Konfiguratinsdatei behob diese Schwierigkeiten. Mit Virtuoso gab es allerdings weiterhin Probleme bezüglich das Datenupload, die wohl bedingt durch verschiedene Versionen sind. Die Probleme wurden durch Upload-Skripte umgangen. Zu beobachten ist, dass die Menge der Daten Virtuoso in einen niedrigen Performance-Bereich bringt.

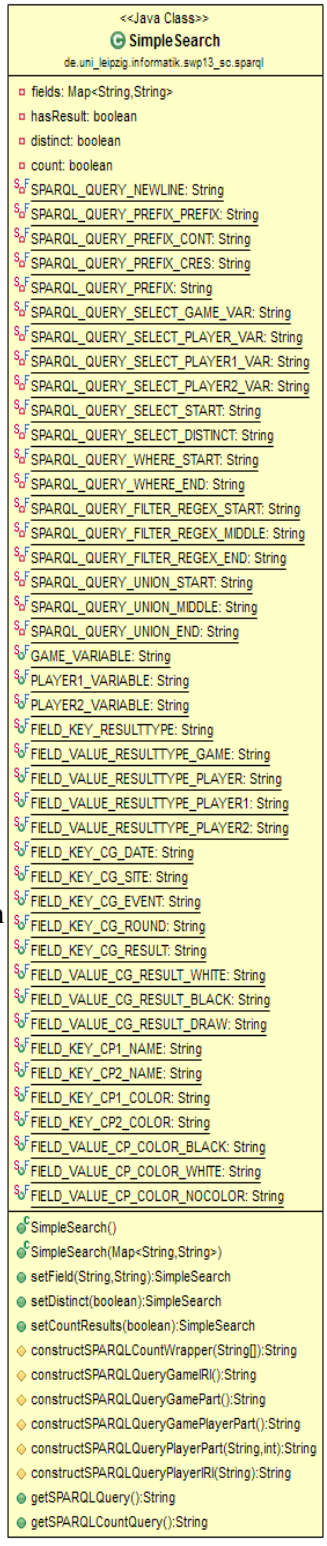
1.K.1 Eröffnungsontologie für die Anzeige von Eröffnungsnamen beim Browsen und Suchen von Eröffnungen

Eine Eröffnungsontologie wurde ähnlich wie die Schachontologie erstellt und in das System integriert, so dass bezüglich dieser Ontologie auch konvertiert werden kann. Dabei wurde sich auf die bekanntesten bzw. häufigsten Eröffnungen beschränkt.

2. Linking mit anderen Datenquellen

2.M.1 Verlinkung mit der Dbpedia

Das Linking mit DBPedia erfolgt mittels LIMES. Dabei wurde eine Konfigurationsdatei entsprechend des Handbuches und der Beispiele erstellt. Das Mapping, welches von dieser Konfigurationsdatei erzeugt wird, wird programmatisch aufgerufen. Zu jeder aufgeführten DBPedia-Resource werden dabei bisher rund 30 Abfragen gestartet um die relevanten Informationen zu erhalten, was sich zeitlich bemerkbar macht. Die hohe Zahl der Anfragen hängt vor allem damit zusammen, dass in der DBPedia verschiedene Attribute die gleichen Informationen bereitstellen, aber selten in einer ähnlichen Belegung vollständig sind.
Die gefundenen Informationen werden nun auf Gleichheit überprüft. Verschiedene Informationen zu einem Attribut werden grundsätzlich alle übernommen. Mit den Resourcen des Semantic Chess Project werden dann die Informationen im *.nt Format gespeichert. Die LinkedData.nt findet sich dabei im Programmverzeichnis und ist manuell in den Virtuoso Triple Store hochzuladen.



2.K.1 Verlinkung mit anderen geeigneten semantischen Wissensbeständen

Es ist zunächst eine Recherche über geeignete Wissensbestände nötig. Eine Anknüpfung könnte möglicherweise auch mit LIMES erfolgen. Die Verlinkung stand aber nicht im Vordergrund des Projektes, so dass auf dieses optionale Ziel verzichtet wurde.

3. Abfrage

3.M.1 Erleichterte Abfrage für Ungelernte

Ein intuitiv bedienbares Suchsystem wurde entwickelt. Die Suchmaske lässt die Suche über verschiedene Attribute zu. Diese werden intern in SPARQL-Code übersetzt. Wird nach mehreren Attributen gesucht, so kommt es intern zu einer Konjunktion der Ergebnisse. Als Ergebnisraum ist es möglich zwischen Spielen und Spielern zu unterscheiden. Zur Generierung der SPARQL-Anfragen wurde auf diverse String-Konstanten zurückgegriffen, die nach dem Baukastenprinzip zusammen gebaut werden können.

3.M.2 Advanced mode mittels eigener SPARQL-Anfragen

SPARQL-Anfragen können ebenfalls direkt über die Benutzeroberfläche gestellt werden. Über eine Verbindung zu dem Datenverwaltungssystem Virtuoso, ist dieses dann in der Lage auf die SPARQL-Anfragen zu reagieren. Alternativ können SPARQL-Anfragen an den Endpoint direkt gerichtet werden: http://pcai042.informatik.uni-leipzig.de:1358/sparql

3.M.3 Statistische Auswertung für Gewinnwahrscheinlichkeiten

Die Auswertung zu den Gewinnwahrscheinlichkeiten für Weiß, Schwarz und Unentschieden erfolgt durch Nutzen der FEN-Notation, die einen bestimmten Schachbrett-Aufbau beschreibt. Dabei werden alle bis zum aktuellen Zug übereinstimmenden Partien mit ihrem Ausgang abgefragt und die Ergebnisse als Wahrscheinlichkeit/relative Häufigkeit ausgegeben.

3.K.1 Abfragen für die Eröffnungsontologie

Mit Vorhandensein der Eröffnungsontologie 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

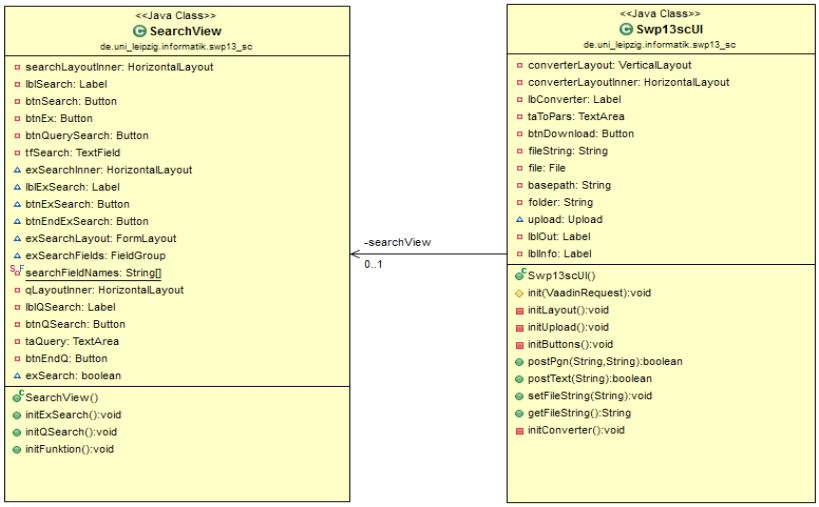
Als Engine wird Carballo verwendet. Die Carballo-Anwendung kann direkt über die grafische Oberfläche aufgerufen werden und stellt einen Teil dieser dar. Sie besteht aus Schachbrett, Figuren, PGN und FEN-Feldern.

4.K.1 Verknüpfung der Schachengine mit der semantischen Wissensbasis
Um die Engine mit der semantischen Wissensbasis 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. Dieses optionale Ziel konnte nicht umgesetzt werden, da die Engine-Integration zu viele technische Probleme mit sich brachte.

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 bringt die verwendete UI der Carballo-Engine mit sich: es ist möglich an einem Computer gegeneinander Schach zu spielen. Das Spiel mit zwei Rechnern über das Internet ist nicht möglich.

5. Weboberfläche



Bei dem Design der Web-Oberfläche wurde vor allem auf Übersichtlichkeit, Einfachheit und Benutzbarkeit geachtet. Für ihre Gestaltung wurde durch Vaadin eine in Java geschriebene WebGUI aufgesetzt und mithilfe des Tomcat-Servers visualisiert. Die GUI umfasst die folgenden Aspekte:

5.M.1 Suchmaske für vereinfachte Abfragen sowie SPARQL-Abfragen
Die Suchmaske besteht aus Eingabefeldern für einfache und erweiterte Suche. Die für beide Suchen dienende Ausgabe erfolgt unter den Eingabefeldern und ist zunächst unsichtbar.

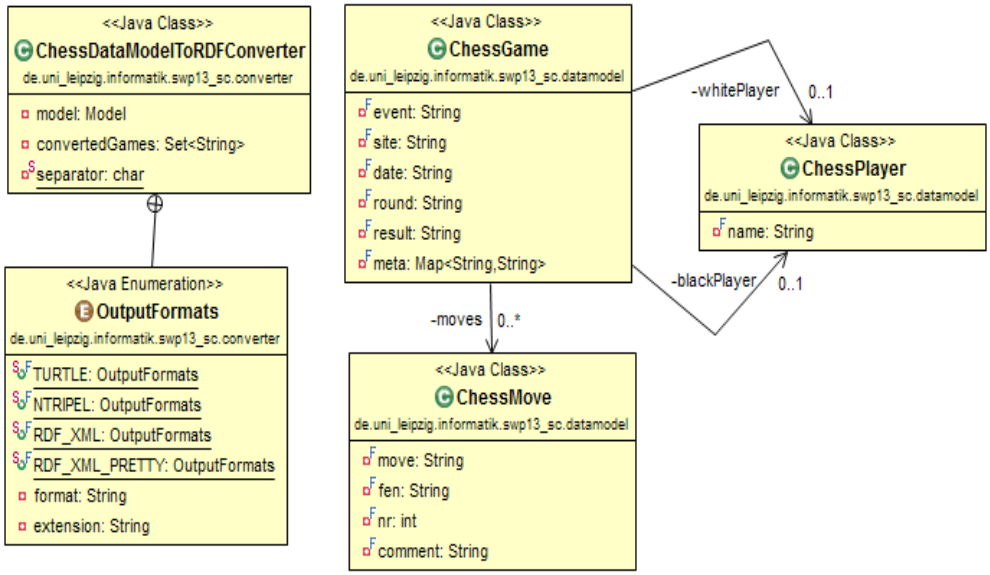
5.M.2 Grafisches Schachbrett mit Figuren, Integration einer Schachengine, Durchspielen von Partien
Eine Engine stellt optisch den Kern der Oberfläche da. Sie lässt sich über den Button GameExplorer aufrufen.

5.M.3 Anzeige der Gewinnwahrscheinlichkeiten der nächsten Züge
Einfache Balkendiagramme geben die Wahrscheinlichkeiten für Sieg/unentschieden/ oder Niederlage an.

5.M.4 Konvertierungs-/Uploadfunktion
Ein Auswahlfeld, sowie ein Bestätigungsbutton stellen die Konvertierungs-/Uploadfunktion bereit.

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

5 Datenmodell



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

FEN (Forsyth-Edwards-Notation)
Kurznotation zur Niederschrift beliebiger Schachbrettstellungen.

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.