M. Herczeg & M. C. Kindsmüller (Hrsg.): Mensch & Computer 2008: Viel Mehr Interaktion.
München: Oldenbourg Verlag, 2008, S. 401-404

ListGraph: Visuelle Analyse von RDF-Daten

Philipp Heim, Deniz Dalli, Jürgen Ziegler

Interaktive Systeme, Universität Duisburg-Essen

Zusammenfassung

Der ListGraph ist ein Tool zur Visualisierung von großen RDF-Datensätzen im Webbrowser. Durch die Kombination von Listen- und Graph-Darstellung werden die jeweiligen Nachteile dieser Darstellungsarten ausgeglichen und dadurch eine verständlichere Visualisierung von RDF-Daten ermöglicht.

1 Einleitung

Das Semantic Web setzt sich immer mehr durch. Dies gilt insbesondere für Bereiche in denen weltweit verteilte große Datenbestände integriert werden müssen. Dazu zählt der Bereich des Requirements-Engineering, in welchem, wie z.B. im *Softwiki*-Projekt[1], Anforderungen semantisch strukturiert werden, um eine bessere Zusammenarbeit von großen und räumlich getrennten Stakeholdergruppen zu ermöglichen (Auer et al. 2006). Dies wird durch die Einordnung der Anforderungen in Ontologien erreicht, in denen zentrale Klassen wie z.B. Anforderung, Stakeholder oder Referenzpunkt definiert sind. Die auf diese Weise semantisch strukturierten Anforderungen werden in RDF abgespeichert, was zu großen RDF-Datensätzen führen kann. Um Menschen den Zugang zu großen RDF-Datensätzen, bzw. zu den darin gespeicherten Informationen zu ermöglichen, bedarf es geeigneter Visualisierungstechniken.

Für die Visualisierung von RDF-Daten existieren diverse Ansätze, welche sich in Graph-basierte Ansätze und Listen-basierte Ansätze unterteilen lassen. Beide haben gewisse Vor- und Nachteile. So eignen sich einerseits Graph-basierte Ansätze im Gegensatz zu Listen-basierten Ansätzen besser, um dem netzwerkartigen Charakter von RDF-Daten Rechnung zu tragen. Andererseits kann die Graph-basierte Visualisierung von großen RDF-Datensätzen mit Tausenden von Knoten und Kanten sehr unübersichtlich werden und dadurch eine für den Menschen verständliche Visualisierung der RDF-Daten behindern. Hier sind Listen-

[1] SoftWiki: BMBF-Verbundprojekt (http://www.softwiki.de).

basierte Ansätze besser geeignet, da sie große Datensätze in übersichtlicherer Weise darstellen können.

Mit dem *ListGraph* stellen wir ein Tool zur Visualisierung von RDF-Daten vor, das beide Ansätze kombiniert und dadurch die jeweiligen Nachteile der einzelnen Ansätze auszugleichen versucht. Der *ListGraph* nutzt die Darstellung als Liste für die Handhabung von großen Informationsmengen und die Darstellung als Graph für die detaillierte Betrachtung weniger ausgewählter Informationen. Dabei startet der Nutzer die Suche nach Informationen indem er, entsprechend seiner Intention, eine bestimmte Klasse von Instanzen auswählt, welche er durchsuchen will. Nach der Wahl dieser Klasse, der *Hauptklasse*, bekommt er alle Instanzen dieser Klasse in einer Liste angezeigt und kann diese nach verschiedenen Kriterien ordnen und durch diverse Filtermethoden einschränken. Auf diese Weise beschränkt der Nutzer die Information bis zu einem Maß, bei dem es ihm möglich wird, besonders interessante Instanzen in der Liste ausfindig zu machen. Dieses Set an ausgewählten Instanzen kann er dann in eine Graph-Darstellung überführen, um sich die netzwerkartige Informationsstruktur explizit anzeigen zu lassen und dadurch ein besseres Verständnis für die Zusammenhänge zu erhalten.

2 Nutzung und Visualisierung des ListGraph

Im Bereich des Requirements-Engineering kann der *ListGraph* beispielsweise eingesetzt werden, um vor dem Einstellen einer neuen Anforderung zu überprüfen, ob zu dieser Thematik bereits Anforderungen existieren. Der Abgleich mit bestehenden Anforderungen kann helfen Konflikte und Überschneidungen frühzeitig zu erkennen. Für eine solche Überprüfung wählt der Nutzer als erstes die Klasse der Anforderungen als *Hauptklasse* und bekommt daraufhin alle Anforderungen in einer Liste angezeigt. Diese Liste kann er dann über Filter beliebig einschränken. Hat er in der Liste interessante Anforderungen ausgemacht, kann er sich diese in einem Graph anzeigen lassen. Dabei kann der Graph durch immer weitere Anforderungen aus unterschiedlich gefilterten Listen iterativ erweitert werden. Die Liste fungiert somit als Pool, um interessante Anforderungen zu isolieren und diese dann genauer im Graphen betrachten zu können. Um ein Verständnis dafür zu bekommen, in welcher Weise die ausgewählten Anforderungen in Beziehung zueinander stehen, werden direkte und indirekte Übereinstimmungen durch benannte Kanten angezeigt.

Die grafische Oberfläche des *ListGraph* setzt sich aus drei Sichten zusammen (Abbildung 1). Mit diesen ist es dem Nutzer möglich, per SPARQL[2] auf RDF-Daten zuzugreifen und diese dann mit Hilfe von Adobe Flex[3] im Browser anzeigen zu lassen. Im Folgenden werden die drei Sichten der *ListGraph*-Oberfläche näher beschreiben.

---

2 SPARQL Protocol and RDF Query Language. W3C-Standard zur Abfrage von RDF-Daten.

3 Adobe FLEX. Entwicklungsframework zum Erstellen von Rich Internet Applications.

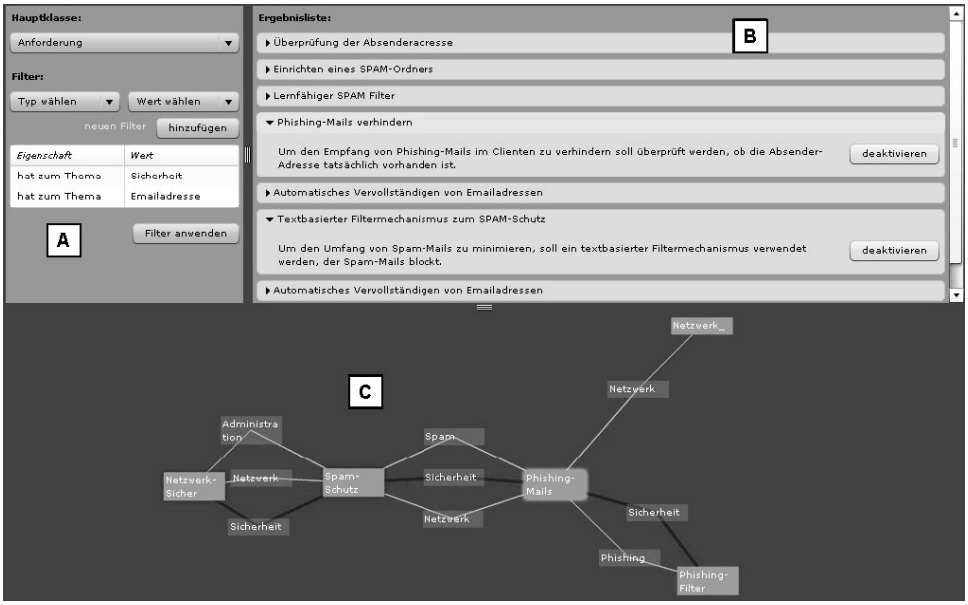

Abbildung 1: Filter-Sicht (A), Listen-Sicht (B) und Graph-Sicht (C)

1. **Filter-Sicht:**

   Ein Filter setzt sich stets aus einer Eigenschaft und einem konkreten Wert für diese Eigenschaft zusammen. Dabei sind nur Eigenschaften der aktuellen *Hauptklasse* als Filtereigenschaft zugelassen. Auf diese Weise kann die Liste aller Anforderungen beispielsweise auf Anforderungen zum Thema „Sicherheit" reduziert werden. Die Eigenschaft wäre dann „hat zum Thema" und der konkrete Wert dementsprechend „Sicherheit". Um die Anzahl der Anforderungen weiter einzuengen, lassen sich mehrere Filter kombinieren (Abbildung 1, A).

2. ***Listen-Sicht*:**
   Die auf diese Weise gefilterte Liste von Anforderungen kann der Nutzer dann nach verschiedenen Kriterien ordnen und die geordnete Liste dann durchsuchen. Beispielsweise können die Anforderungen nach ihrem Entstehungsdatum angeordnet werden. Findet der Nutzer in der Liste eine interessante Anforderung, kann er diese in der Listenansicht aufklappen, um die komplette Anforderung lesen zu können (Abbildung 1, B). Der Nutzer kann die Anforderung interaktiv, durch Klicken des Aktivieren-Buttons, als neuen Knoten in die Graph-Darstellung übernehmen, um sie dort im Zusammenhang betrachten zu können.

3. ***Graph-Sicht*:**
   Die Visualisierung als Graph soll dem Nutzer ein besseres Verständnis sowohl für die direkten, als auch für die indirekten Zusammenhänge innerhalb des ausgewählten Sets von Anforderungen ermöglichen. Um das Verständnis für die direkten Zusammenhänge innerhalb des Sets zu fördern, werden die ausgewählten Anforderungen entsprechend ihrer

Ähnlichkeit über Kanten miteinander verbunden. Anforderungen sind sich ähnlich, wenn sie gleiche Eigenschaften teilen. Ähnliche Anforderungen werden somit durch mehrere Kanten miteinander verbunden und dadurch im Force-Directed-Layout (Fruchterman & Reingold 1991) des Graphen nahe beieinander positioniert.

Soweit kein direkter Zusammenhang zwischen einer neu aktivierten Anforderung und den bereits bestehenden Anforderungen im Graph vorliegt, wird nach indirekten Zusammenhängen gesucht. Für einen indirekten Zusammenhang benötigt man mindestens eine zusätzliche Anforderung, welche die neu aktivierte Anforderung mit den bereits bestehenden Anforderungen durch jeweils gleiche Eigenschaften verknüpfen kann. Dabei wird immer nur derjenige indirekte Zusammenhang dargestellt, der die meisten gleichen Eigenschaften aufweist und dafür die wenigsten zusätzlichen Anforderungen benötigt.

Dadurch wird der Nutzer befähigt, direkt vom Graph ablesen zu können, welche Eigenschaften die ausgewählten Anforderungen gemeinsam haben und welche indirekten Zusammenhänge bestehen. Zusätzlich können alle Kanten einer Eigenschaft farbig hervorgehoben werden, um eine bestimmte Eigenschaft besser nach verfolgen zu können (Abbildung 1, C).

3 Zusammenfassung

Für die Anzeige der Informationen verwendet der *ListGraph* sowohl eine Listen- als auch eine Graph-Visualisierung. Die Listen-Visualisierung erlaubt die Handhabung großer Datenmengen in übersichtlicher Weise, während die Graph-Visualisierung die Darstellung komplexer direkter und indirekter Zusammenhänge innerhalb eines Sets ausgewählter Instanzen ermöglicht. Durch die Kombination beider Darstellungsarten können die jeweiligen Nachteile einer Listen- und einer Graph-Visualisierung ausgeglichen werden. Durch die Möglichkeit bestimmte Instanzen aus unterschiedlichen Listen auszuwählen und deren Beziehungen auf Grund von gemeinsamen Eigenschaften zu visualisieren, wird das Auffinden von Überschneidungen, Konflikten und Abhängigkeiten zwischen Instanzen unterstützt. Der *ListGraph* bietet somit eine verständlichere Visualisierung von RDF-Daten und verhilft dadurch auch zu einer besseren Analyse der in diesen Daten enthaltenen Informationen.

Literaturverzeichnis

Auer, S., Riechert, T. & Fähnrich, K.-P. (2006). SoftWiki – Agiles Requirements-Engineering für Softwareprojekte mit einer großen Anzahl verteilter Stakeholder. In *Workshop GeNeMe 06*.

Card, S., Mackinlay, J. & Shneiderman, B. (1999), *Readings in Information Visualization: Using Vision to Think*, San Francisco, CA: Morgan Kaufmann Publishers.

Fruchterman, T. & Reingold, E. (1991). Graph drawing by force-directed placement. In *Software-Practice & Experience*, S.1129-1164.