



Universitat de Lleida

Interacting with Semantic Web Data through an Automatic Information Architecture

Josep Maria Brunetti Fernández

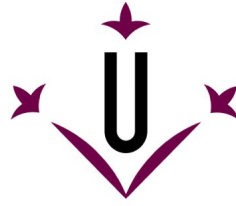
Dipòsit Legal: L.313-2014

<http://hdl.handle.net/10803/131223>



Interacting with semantic web data through an automatic information architecture està subjecte a una llicència de [Reconeixement-NoComercial-CompartirIgual 3.0 No adaptada de Creative Commons](https://creativecommons.org/licenses/by-nc-sa/3.0/)

(c) 2013, Josep Maria Brunetti Fernández



Universitat de Lleida
Escola Politècnica Superior

Interacting with Semantic Web Data through an Automatic Information Architecture

by

Josep Maria Brunetti Fernández

Thesis submitted to the University of Lleida in fulfillment
of the requirements for the degree of Doctor in Computer Science

Under supervision of PhD Roberto García González
Lleida, December 2013

Acknowledgments

Voldria mostrar el meu agraïment a totes aquelles persones que han col·laborat d'una manera o altra amb aquesta tesi. I després d'escriure tantes pàgines en anglès, voldria fer-ho en català. Al cap i a la fi, si en aquesta memòria hi ha una petita part on puc expressar lliurement allò que sento, és aquí; i no tinc millor manera de fer-ho que en català, perquè és la meva llengua materna i és amb la que millor m'expresso. Tot i que alguns es capfiquin en canviar-li el nom o intentin reduir-ne el seu ús, som moltes persones les que parlem en català i seguirem fent-ho. Només demanem que es respecti com qualsevol altra llengua.

En primer lloc, el meu més sincer agraïment al Roberto García, director d'aquesta tesi. Gràcies per donar-me aquesta possibilitat, confiar en mi, aconsellar-me, guiar-me i a la vegada donar-me llibertat per tirar endavant les meves idees. Igualment, voldria donar les gràcies a tots aquells amb els qui he treballat durant el desenvolupament d'aquesta tesi i que hi han contribuït: Juan Manuel Gimeno, Rosa Gil, Antonio López i José Luís González.

Voldria donar les gràcies de forma especial a la Llúcia Masip, companya de doctorat al grup i amiga, amb qui he compartit viatges, congressos, alegries i penes durant aquests anys. Sempre que he necessitat res o he tingut algun dubte amb algun tràmit, només he hagut de trucar a la porta del davant i preguntar. Sense tu tot això hauria estat molt més difícil.

A la resta de professors, estudiants i companys del grup GRIHO, gràcies per acollir-me tan bé al grup des del primer dia i fer que em sentís com a casa. La llista de persones amb les que he compartit el dia a dia és molt llarga i no em voldria deixar a ningú. Sobretot, gràcies per ajudar-me en els moments difícils. No només he tingut companys de feina, sinó amics.

A nivell econòmic, aquesta tesi ha estat possible gràcies a una beca predoctoral que em va atorgar la Universitat de Lleida. Igualment, he gaudit d'altres ajuts de la universitat que han permès finançar parcialment l'assistència a congressos i l'estada predoctoral que vaig realitzar a Leipzig. En aquest sentit, també voldria agrair a l'empresa GFT Group i en especial al Carlos Lozano el seu interès pel nostre treball i que ens convidessin a assistir al CEBIT de Hannover. Tan de bo d'aquestes col·laboracions en puguin sortir projectes interessants.

Igualment, part d'aquest treball així com l'assistència a congressos ha comptat amb el suport del Ministerio de Ciencia e Innovación a través dels projectes OMediaDis (Open Platform for Multichannel Content Distribution Management, TIN2008-06228) i InDAGuS (Infrastructures for Sustainable Open Government Data with Geospatial Features, TIN2012-37826-C02). Tot i aquest suport, sóc una de les moltes persones que s'han vist afectades per les retallades, la reducció de sous o l'augment del preu de les matricules de la universitat. Des d'aquí, mostrar el meu rebuig total a les polítiques que s'estan duent a terme en ensenyament i investigació en aquest país. És una llastima que el treball de molta gent quedi pràcticament en un no-res per la falta de finançament. Tan de bo ben aviat es pugui canviar aquesta tendència.

Vaig tenir l'oportunitat de fer una estada de tres mesos al grup AKSW de Leipzig on vaig poder treballar amb altres investigadors i estudiants en temes relacionats amb el d'aquesta tesis. Voldria agrair a Sören Auer la seva dedicació i col·laboració, especialment en relació a la visualització de dades enllaçades. Estic molt agraït pel temps que vaig passar a Leipzig i per tota la gent que hi vaig conèixer, sens dubte va ser una gran experiència. També vull donar les gràcies a Jakub Klímek i Martin Nežaský, de la Charles University de Praga, amb els qui també hem col·laborat en aquesta temàtica.

Durant aquest temps per la universitat he compartit esmorzar principalment amb dos bons amics. Al Francesc Guitart i Cèsar Fernández, gràcies per tot el temps i activitats que compartiu amb mi. Retrobar-me amb tot allò que m'agrada ha estat una de les sorpreses més agradables d'aquests últims anys. També voldria mostrar el meu agraïment a la Meritxell Pérez, amb qui he compartit el poc temps lliure que he tingut i m'ha donat suport en aquests últims mesos més estressants.

Finalment, voldria donar gràcies als meus pares, germans i família. Són molts anys els que porto davant d'una pantalla i tot i no entendre exactament a què em dedico, sempre ho heu respectat i m'heu donat suport.

Difícilment es poden resumir més de tres anys en un full de paper. És molt difícil donar les gràcies a tanta gent i per tantes coses; per molt que escrivís, mai seria suficient. A tots aquells que heu contribuït d'una manera o altra en aquesta tesis o que heu estat al meu costat durant tots aquests anys, moltes gràcies de tot cor.

M'enduc un munt d'amics i una pila d'experiències innoblidables: els aniversaris cel·lebrats a GRIHO, els kebabs del dijous, esmorzars i batalletes, viatges, congressos, ciutats que he conegut, una petita estada a Alemanya... En el fons, són aquestes petites coses les que donen sentit a tot plegat i que fan aquests anys hagin estat, simplement, genials.

I would like to acknowledge the people who somehow contributed to this work. First of all, I would like to thank my supervisor Dr. Roberto García for guiding me during this work and granting me the freedom to develop my research ideas. I would also like to express my gratitude to all the members of the GRIHO Group of the Universitat de Lleida whom I worked with and shared friendship.

Financially, I was supported by a PhD grant from the Universitat de Lleida. This work was also partially supported by Spanish Ministry of Science and Innovation through research projects OMediaDis (Open Platform for Multichannel Content Distribution Management, TIN2008-06228) and InDAGuS (Infrastructures for Sustainable Open Government Data with Geospatial Features, TIN2012-37826-C02).

I had the opportunity to work for three months within the AKSW group at the University of Leipzig. I would like to thank Dr. Sören Auer, head of the group, who inspired me with ideas and guided me in some parts of these work, especially in the Linked Data Visualization Model. I am grateful for my time in Leipzig where I met all the people I met there.

Finally, I would also like to thank Jakub Klímek and Martin Nežaský, from the Charles University in Prague. Our collaboration with them led to new ideas in the research topic of visualizing Linked Data.

The amount of semantic data available in the Web is rapidly increasing. The proliferation of Linked Open Data and other data publishing initiatives has increased the amount of data available for analysis and reuse. The potential of this vast amount of data is enormous but it results in the need for handling the Information Overload phenomenon. In most cases it is very difficult for users to explore and use this data, especially for those without experience with Semantic Web technologies. This is due to the lack of tools designed for end-users and the prevalence of specialised browsers that require complex queries to be formed and intimate knowledge on how datasets are structured.

Our contribution in order to solve this problem is applying the Visual Information-Seeking Mantra: “Overview first, zoom and filter, then details-on-demand”, which is implemented using Information Architecture components users are already familiar with. Information Architecture is the discipline that organizes and labels the information on websites, analyzing the contents, organizing web pages and designing the navigation systems. This thesis offers algorithms and methods to automatically generate and drive the information architecture components for websites based on Semantic Web data. These components are automatically generated from data and ontologies, what makes the proposed approach generic and scalable.

First of all, users can obtain an overview of the dataset through different components: navigation menus, site map, site index or treemap. These components are useful for obtaining a broad view of the datasets, the main types of things they describe and how they are relate. They serve as a starting point for navigation. Once the user has overviewed the dataset and detected the types of entities he is interested in, it is time to explore them. Faceted navigation and pivoting allow users to perform an exploratory search, see the most significant properties for different kinds of items and filter them based on their attributes and how they are connected to other entities. Breadcrumbs provide context information to guide users to their target, showing them the path taken so far and allowing them to go back to previous steps in the exploration process. Finally, once users have selected those resources they are interested in, they can obtain a detailed view of them and see specific visualizations such as maps, timelines and charts, depending on their kind and the attributes and properties describing.

This approach has been implemented and tested in *Rhizomer*, a tool capable of publishing Semantic Web datasets while facilitating user awareness of the published content. *Rhizomer* includes Information Architecture components automatically generated from the underlying semantic data and ontologies that support the overview, zoom and filter and details on demand tasks. The previous contributions have been validated with end users as part of a User Centred Design development process. Evaluations during an iterative development process and a novel quality in use model for Semantic Web exploration tools have motivated and guided the introduction of new features, detected user experience issues and validated the approach together with the resulting implementations, even when compared with similar tools.

This thesis proves that the process of creating the Information Architecture of a website based on semantic data can be automated and generalised to different schemas. Through automatic Information Architecture components, even lay-users are capable of building complex queries while they interact with Semantic Web data without requiring to learn complicated technologies or knowing the vocabularies used in the explored datasets.

La quantitat de dades semàntiques disponibles a la Web està augmentant ràpidament. La proliferació d'iniciatives de publicació de dades com Linked Open Data ha incrementat la quantitat de dades disponibles per analitzar i reutilitzar. El potencial de tal quantitat de dades és enorme, però és necessari afrontar el fenomen de la sobrecàrrega d'informació que produeix als usuaris. En molts casos, és molt difícil pels usuaris explorar i utilitzar aquestes dades, especialment quan no tenen experiència en tecnologies de Web Semàntica. Això és degut a l'escassetat d'eines pensades per a usuaris finals i al predomini de navegadors especialitzats que requereixen formular consultes complexes i un ampli coneixement de com estan estructurats els conjunts de dades.

La nostra contribució per a solventar aquest problema és aplicar el mantra de la cerca visual d'informació (*Visual Information-Seeking Mantra*): "Primer una visió general, enfocar i filtrar, després detalls sota demanda". Les tasques descrites en el mantra s'han implementat utilitzant components típics de l'Arquitectura de la Informació i que són coneguts per la majoria d'usuaris. L'Arquitectura de la Informació és la disciplina que organitza i etiqueta la informació en llocs web, analitzant-ne els continguts, organitzant-ne les pàgines web i dissenyant-ne els sistemes de navegació. Aquesta tesi ofereix algorismes i mètodes per a generar automàticament components de l'Arquitectura de la Informació per a llocs web basats en dades semàntiques. Aquests components es generen automàticament a partir de les dades i ontologies, fent que aquesta proposta sigui genèrica i escalable.

En primer lloc, els usuaris poden obtenir una visió global del conjunt de dades a través de diferents components: menús de navegació, mapa del lloc web, índex del lloc web o treemap. Aquests components permeten tenir una visió àmplia de les dades disponibles, els principals tipus i les relacions entre ells. Aquests components serveixen de punt de partida a l'hora de navegar per les dades. Un cop els usuaris han obtingut una visió general del conjunt de dades i detectat els principals tipus d'entitats que els interessen, és hora d'explorar-los. La navegació per facetes i el pivotat permeten als usuaris fer una cerca exploratòria, veure les propietats més rellevants per diferents tipus de dades i filtrar-les. Les molles de pa (*breadcrumbs*) proporcionen informació contextual per a guiar els usuaris fins al seu objectiu, els permeten veure el camí que han seguit i retornar a pàgines visitades anteriorment. Finalment, quan els usuaris han sel·leccionat aquells recursos d'interés, poden obtenir més detalls sobre ells i visualitzar-los en mapes, línies temporals o gràfiques, depenent del tipus de dades que descriuen.

Aquesta proposta s'ha implementat i validat a *Rhizomer*, una eina capaç de publicar conjunts de dades basats en Web Semàntica i que facilita als usuaris entendre-les i interactuar amb elles. Les contribucions descrites prèviament s'han avaluat amb usuaris finals com a part d'un Disseny Centrat en l'Usuari. Les avaluacions durant el desenvolupament iteratiu han motivat la introducció de noves funcionalitats, han permès detectar problemes d'usabilitat i validar els requeriments. A més a més, el model de qualitat en l'ús per a eines d'exploració de la Web Semàntica que hem proposat ha permès comparar la nostra implementació amb eines similars.

En aquesta tesis es demostra que el procés de generar l'Arquitectura de la Informació d'un lloc web basat en dades semàntiques es pot automatitzar i generalitzar per a diferents esquemes. A través de components de l'Arquitectura de la Informació automàtics, els usuaris són capaços de construir consultes complexes mentre interactuen amb les dades, sense necessitat d'aprendre tecnologies complicades o conèixer els vocabularis utilitzats en els conjunts de dades que s'estan explorant.

La cantidad de datos semánticos disponibles en la Web está aumentando rápidamente. La proliferación de iniciativas de publicación de datos como Linked Open Data ha incrementado la cantidad de datos disponibles para su análisis y reuso. El potencial de tal cantidad de datos es enorme, pero en muchos casos es necesario afrontar el fenómeno de la sobrecarga de información que produce a los usuarios. En muchos casos, es muy difícil para los usuarios explorar y utilizar estos datos, especialmente cuando no tienen experiencia en tecnologías de Web Semántica. Esto es debido a la escasez de herramientas diseñadas para usuarios finales y al predominio de navegadores especializados que requieren formular consultas complejas y un amplio conocimiento de la estructura de los datos.

Nuestra contribución para solventar este problema se basa en aplicar el mantra de la búsqueda visual de información (*Visual Information-Seeking Mantra*): “Primero una visión general, enfocar y filtrar, después detalles bajo demanda”. Éstas tareas descritas en el mantra se han implementado utilizando componentes típicos de la Arquitectura de la Información y que son conocidos por la mayoría de usuarios. La Arquitectura de la Información es la disciplina que organiza y etiqueta la información en sitios web, analizando sus contenidos, organizando sus páginas web y diseñando sus sistemas de navegación. Esta tesis ofrece algoritmos y métodos para generar automáticamente componentes de la Arquitectura de la Información para sitios web basados en datos semánticos. Estos componentes se generan automáticamente a partir de los datos y ontologías, haciendo que esta propuesta sea genérica y escalable.

Primero, los usuarios pueden obtener una visión global del conjunto de datos gracias a distintos componentes: menú de navegación, mapa del sitio web, índice del sitio web o treemap. Estos componentes permiten tener una visión amplia de los datos disponibles, los principales tipos y las relaciones entre ellos. Estos componentes sirven de punto de partida para navegar por los datos. Cuando los usuarios han obtenido una visión general del conjunto de datos y detectado los principales tipos de entidades que les interesan, es hora de explorarlos. La navegación por facetas y el pivotado permiten a los usuarios realizar una búsqueda exploratoria, ver las propiedades más relevantes para distintos tipos de datos y filtrarlos. Las migas de pan (*breadcrumbs*) proporcionan información contextual para guiar a los usuarios hasta su objetivo, les permiten ver el camino que han seguido y volver a páginas visitadas con anterioridad. Finalmente, cuando los usuarios han seleccionado aquellos recursos de interés, pueden obtener más detalles sobre ellos y visualizarlos en mapas, líneas temporales o gráficas, dependiendo del tipo de datos que describan.

Esta propuesta se ha implementado y validado en *Rhizomer*, una herramienta capaz de publicar conjuntos de datos basados en Web Semántica y que facilita a los usuarios su comprensión y la interacción con ellos. Las contribuciones descritas previamente se han evaluado con usuarios finales como parte de un Diseño Centrado en el Usuario. Las evaluaciones durante el desarrollo iterativo han motivado la introducción de nuevas funcionalidades, han permitido detectar problemas de usabilidad y validar los requisitos. Además, el modelo de calidad en el uso que hemos propuesto ha permitido comparar nuestra implementación con herramientas similares.

En esta tesis se demuestra que el proceso de generar la Arquitectura de la Información de un sitio web basado en datos semánticos puede automatizarse y generalizarse para distintos esquemas. Gracias a estos componentes automáticos de la Arquitectura de la Información, los usuarios son capaces de construir consultas complejas mientras interactúan con los datos, sin necesidad de aprender tecnologías complicadas o conocer los vocabularios utilizados en los conjuntos de datos que se están explorando.

| | |
|---|-----------|
| List of Figures | 15 |
| List of Tables | 17 |
| I Prelude | 21 |
| 1. Introduction | 23 |
| 1.1. Motivation | 23 |
| 1.2. Problem statement | 24 |
| 1.2.1. Defining End Users | 24 |
| 1.2.2. Human-Semantic Web Interaction | 26 |
| 1.3. Hypothesis | 27 |
| 1.4. Contributions | 28 |
| 1.5. Outline | 29 |
| II Background | 31 |
| 2. Standards and Technologies of the Semantic Web | 33 |
| 2.1. The Semantic Web | 33 |
| 2.1.1. From a Web of documents to a Web of data | 34 |
| 2.1.2. RDF | 35 |
| 2.1.3. RDF Schema | 36 |
| 2.1.4. Ontologies and OWL | 37 |
| 2.1.5. XML and XML Schema | 37 |
| 2.1.6. SPARQL | 37 |
| 2.2. Linked Data | 39 |
| 2.3. Linked Open Data | 40 |
| 3. Related Work | 43 |
| 3.1. Semantic Web browsers | 43 |
| 3.1.1. Text-based browsers | 43 |
| 3.1.2. Graph-based browsers | 47 |
| 3.1.3. Browsers Supporting Pivoting | 48 |
| 3.2. Ontology Visualization | 50 |
| 3.3. CMS and Semantic Wikis | 51 |
| 3.4. Summary | 51 |
| 3.5. Rhizomer | 53 |

| | | |
|------------|--|-----------|
| III | Preparation | 55 |
| 4. | Approach | 57 |
| 4.1. | Information Architecture | 57 |
| 4.2. | Tasks for data analysis | 59 |
| 4.2.1. | Overview | 60 |
| 4.2.1.1. | Navigation menus | 60 |
| 4.2.1.2. | HTML Site maps | 61 |
| 4.2.1.3. | Site index | 61 |
| 4.2.1.4. | Treemap | 61 |
| 4.2.2. | Zoom & Filter | 61 |
| 4.2.2.1. | Faceted navigation | 62 |
| 4.2.3. | Details-on-demand | 62 |
| 4.2.3.1. | RDF representation and visualization | 63 |
| 4.2.4. | Relate | 64 |
| 4.2.4.1. | Links to related resources | 64 |
| 4.2.4.2. | Set-based browsing | 64 |
| 4.2.5. | History | 64 |
| 4.2.5.1. | Breadcrumbs | 64 |
| 4.2.6. | Extract | 65 |
| 4.2.6.1. | Bookmarks | 66 |
| 5. | Methodology | 67 |
| 5.1. | MPlu+a | 67 |
| 5.1.1. | Overview | 67 |
| 5.1.2. | User-Centered Design | 68 |
| 5.1.2.1. | Usability | 68 |
| 5.1.2.2. | Accessibility | 69 |
| 5.1.3. | Software engineering | 69 |
| 5.1.3.1. | Requirements analysis | 69 |
| 5.1.3.2. | Design | 70 |
| 5.1.3.3. | Implementation | 70 |
| 5.1.3.4. | Launch | 70 |
| 5.1.4. | Prototyping | 71 |
| 5.1.5. | Evaluation | 71 |
| 5.2. | SWET-QUM | 72 |
| 5.2.1. | The concept of quality | 73 |
| 5.2.2. | Quality in Use for SWETs | 74 |
| 5.2.2.1. | Effectiveness | 75 |
| 5.2.2.2. | Efficiency | 76 |
| 5.2.2.3. | Context Coverage | 77 |
| 5.2.2.4. | Satisfaction | 78 |
| 5.3. | Development and evaluation process | 79 |
| 5.3.1. | Pre-test | 79 |
| 5.3.2. | Test | 80 |
| 5.3.3. | Post-test | 80 |
| 5.3.4. | Reports | 80 |

| | |
|---|------------|
| IV Contribution | 81 |
| 6. Automatic Information Architecture Generation Methods | 83 |
| 6.1. Overview generation and storage | 84 |
| 6.1.1. Algorithm to generate navigation menus | 85 |
| 6.1.2. Revising the algorithm | 88 |
| 6.2. Facet discovery and ranking | 90 |
| 6.2.1. Approaches to facet ranking | 91 |
| 6.2.1.1. Frequency-based ranking | 91 |
| 6.2.1.2. Set-cover ranking | 91 |
| 6.2.1.3. Metric-based ranking | 91 |
| 6.2.2. Experimenting with metric-based ranking | 92 |
| 6.2.3. Descriptive facet ranking | 93 |
| 6.2.3.1. Metrics proposed | 94 |
| 6.3. Linked Data Visualization Model | 98 |
| 6.3.1. Overview of LDVM | 98 |
| 6.3.2. LDVM Stages | 100 |
| 6.3.3. Formalization and compatibility | 102 |
| 6.3.4. Implementation | 103 |
| 6.3.4.1. LODVisualization | 103 |
| 6.3.4.2. Rhizomer | 105 |
| 6.3.5. Evaluation | 107 |
| 7. Iterative User Interface Development | 109 |
| 7.1. Iteration 1 | 110 |
| 7.1.1. Requirements analysis | 110 |
| 7.1.1.1. Functional requirements | 110 |
| 7.1.1.2. Non-functional requirements | 110 |
| 7.1.2. Design | 110 |
| 7.1.3. Implementation | 112 |
| 7.1.3.1. Navigation menus | 112 |
| 7.1.3.2. Facets | 112 |
| 7.1.3.3. Breadcrumbs | 114 |
| 7.1.4. Prototyping | 115 |
| 7.1.5. Evaluation | 115 |
| 7.1.5.1. Experimental Design | 116 |
| 7.1.5.2. Tasks | 116 |
| 7.1.5.3. Usability metrics | 117 |
| 7.1.5.4. Results and discussion | 117 |
| 7.1.5.5. Conclusions and proposals | 119 |
| 7.2. Iteration 2 | 120 |
| 7.2.1. Requirements analysis | 120 |
| 7.2.2. Design and implementation | 121 |
| 7.2.2.1. Pivoting in facets | 121 |
| 7.2.2.2. Literal breadcrumbs | 124 |
| 7.2.2.3. Labels | 125 |
| 7.2.3. Prototyping | 125 |
| 7.2.4. Evaluation | 125 |
| 7.2.4.1. Experimental Design | 126 |

CONTENTS

| | | |
|-----------|---|------------|
| 7.2.4.2. | Tasks | 127 |
| 7.2.4.3. | Usability metrics | 127 |
| 7.2.4.4. | Results and discussion | 128 |
| 7.2.4.5. | Conclusions and proposals | 135 |
| 7.3. | Iteration 3 | 137 |
| 7.3.1. | Requirements analysis | 137 |
| 7.3.2. | Design and implementation | 137 |
| 7.3.2.1. | Navigation menus | 137 |
| 7.3.2.2. | HTML site maps | 137 |
| 7.3.2.3. | Site index | 139 |
| 7.3.2.4. | Treemap | 140 |
| 7.3.3. | Evaluation | 140 |
| 7.3.3.1. | Experimental Design | 141 |
| 7.3.3.2. | Tasks | 141 |
| 7.3.3.3. | Usability metrics | 142 |
| 7.3.3.4. | Results and Discussion | 143 |
| 7.3.3.5. | Conclusions and proposals | 145 |
| 7.4. | Iteration 4 | 147 |
| 7.4.1. | Requirements analysis | 147 |
| 7.4.2. | Design and implementation | 147 |
| 7.4.2.1. | Facets re-design | 147 |
| 7.4.2.2. | Encouraging pivoting | 148 |
| 7.4.2.3. | New breadcrumbs design | 149 |
| 7.4.2.4. | Paginating and ordering results | 150 |
| 7.4.3. | Evaluation | 151 |
| 7.4.3.1. | Experimental Design | 151 |
| 7.4.3.2. | Tasks | 151 |
| 7.4.3.3. | Usability metrics | 152 |
| 7.4.3.4. | Results and discussion | 152 |
| 7.4.3.5. | Conclusions and proposals | 157 |
| 7.5. | Iteration 5 | 159 |
| 7.5.1. | Requirements analysis | 159 |
| 7.5.2. | Design and implementation | 160 |
| 7.5.2.1. | Keyword search | 160 |
| 7.5.2.2. | Facet widgets | 162 |
| 7.5.2.3. | Transitions in pivoting | 162 |
| | V Conclusion | 165 |
| 8. | Conclusions and Future Work | 167 |
| 8.1. | Conclusions | 167 |
| 8.2. | Publications | 171 |
| 8.3. | Future Work | 174 |
| A. | User Evaluation Documents | 177 |
| | Bibliography | 181 |

List of Figures

| | |
|--|-----|
| 2.1. The Semantic Web layer cake, from http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#(24) | 34 |
| 2.2. RDF graph describing Dr. Eric Miller, from http://www.w3.org/TR/2004/REC-rdf-primer-20040210/ | 35 |
| 2.3. Linked Open Data (LOD) cloud, from http://lod-cloud.net/ | 41 |
| 3.1. Tabulator RDF browser | 44 |
| 3.2. Exhibit interface | 45 |
| 3.3. DBpedia Faceted Browser | 46 |
| 3.4. Facet columns in mSpace Faceted Browser | 47 |
| 3.5. RDF-gravity interface | 48 |
| 3.6. Visor interface | 49 |
| 3.7. Rhizomer architecture overview | 53 |
| 4.1. Faceted browsing in eBay | 63 |
| 4.2. Faceted browsing in Amazon | 63 |
| 5.1. MPlu+a organization, from http://www.grihohcitoools.udl.cat/mpiu | 67 |
| 5.2. Quality of a software product | 74 |
| 5.3. Quality in use factors | 74 |
| 6.1. Generating a navigation submenu for DBpedia Species with 7 slots (left original, right result) | 87 |
| 6.2. Navigation menu generated for the DBpedia | 88 |
| 6.3. High level overview of the Linked Data Visualization Model. | 99 |
| 6.4. Linked Data Visualization Model ecosystem | 101 |
| 6.5. High-level LODVisualization architecture. | 104 |
| 6.6. Map visualization. | 106 |
| 6.7. Chart visualization. | 106 |
| 7.1. Paper prototype | 111 |
| 7.2. Automatic facets for the http://dbpedia.org/ontology/Film class | 113 |
| 7.3. Software prototype | 115 |
| 7.4. Set-based browsing through pivoting | 124 |
| 7.5. Pivoting enhancements | 126 |
| 7.6. Statistical analysis, iteration 2 | 130 |
| 7.7. Rhizomer heat map, Iteration 2 | 131 |
| 7.8. Virtuoso Faceted Browser heat map | 132 |
| 7.9. SPARQL heat map | 132 |
| 7.10. Post task satisfaction measures for iteration 2 | 134 |
| 7.11. Post test satisfaction measures for iteration 2 | 135 |
| 7.12. Summarized site map | 138 |

LIST OF FIGURES

| | |
|---|-----|
| 7.13. Full site map | 138 |
| 7.14. Site index implementation | 139 |
| 7.15. Treemap of the DBpedia dataset | 140 |
| 7.16. Treemap of the DBpedia dataset (2) | 141 |
| 7.17. Average task completion times | 143 |
| 7.18. Improvements in pivoting in iteration 4. | 148 |
| 7.19. Experimental breadcrumbs including location, path and attributes. | 149 |
| 7.20. Pagination and sorting components | 150 |
| 7.21. Rhizomer heat map, Iteration 4, user group A | 154 |
| 7.22. Rhizomer heat map, Iteration 4, user group B | 155 |
| 7.23. Post task satisfaction measures for iteration 4, task 2 | 157 |
| 7.24. Post task satisfaction measures for iteration 4, task 3 | 157 |
| 7.25. Post test satisfaction measures for iteration 4 | 158 |
| 7.26. Keyword search with results, autocomplete widget and type filtering | 160 |
| 7.27. Resource describing Woody Allen and pivoting options | 161 |
| 7.28. Facet widgets for ordinal and temporal data | 163 |

List of Tables

| | |
|--|-----|
| 1.1. Personas illustrating the intended users | 25 |
| 3.1. Comparison of Semantic Web data exploration tools | 52 |
| 4.1. Tasks for data analysis, interaction patterns and Information Architecture components | 59 |
| 6.1. Automatic facet ranking for the class http://dbpedia.org/ontology/Ship . | 96 |
| 6.2. Generic visualization data types. | 101 |
| 6.3. Evaluation results summary: execution time for 10 combinations of datasets, data extractions and visualization configurations. | 107 |
| 6.4. Timing for each transformation: data transformation, visual transformation and visual mapping transformation. | 108 |
| 7.1. Evaluation results for iteration 1: effectiveness and efficiency | 118 |
| 7.2. Evaluation results for iteration 1: context coverage | 118 |
| 7.3. Comparison with previous evaluation | 128 |
| 7.4. Evaluation results for iteration 2: effectiveness and efficiency | 129 |
| 7.5. Evaluation results for iteration 2: UI component effectiveness and efficiency . . | 131 |
| 7.6. Evaluation results for the iteration 2: context coverage | 133 |
| 7.7. User satisfaction questionnaire for navigation menus (NM), site map (SM), treemap (TR) and site index (SI) | 144 |
| 7.8. Comparison with iteration 2 | 153 |
| 7.9. Evaluation results for iteration 4: UI component effectiveness and efficiency . . | 154 |
| 7.10. Evaluation results for the iteration 4: context coverage | 156 |
| 7.11. Special facet widgets | 162 |

List of Examples

| | | |
|-----|---|-----|
| 1. | (RDF/XML Syntax) | 36 |
| 2. | Structure of a SPARQL query | 38 |
| 3. | SPARQL query with OPTIONAL clause, from http://jena.apache.org/tutorials/sparql.html | 38 |
| 4. | SPARQL query results | 39 |
| 5. | SPARQL query with FILTER clause, from http://jena.apache.org/tutorials/sparql.html | 39 |
| 6. | SPARQL query to obtain the root classes | 84 |
| 7. | SPARQL query to obtain direct subclasses for a given class | 84 |
| 8. | SPARQL query to count the number of instances of each instantiated class | 84 |
| 9. | Part of the VoID description for the DBpedia dataset | 85 |
| 10. | Overview of the navigation menu generation algorithm | 86 |
| 11. | SPARQL query to count the number of properties for a <CLASS> | 89 |
| 12. | Revised navigation menus algorithm | 90 |
| 13. | SPARQL query to obtain values and counts for a <CLASS> and <PROPERTY> | 93 |
| 14. | SPARQL query to obtain all properties for a <CLASS> | 96 |
| 15. | SPARQL query to obtain ontology properties for a <CLASS> | 97 |
| 16. | SPARQL query to obtain the number of instances of a <CLASS> that have a concrete <PROPERTY> | 97 |
| 17. | SPARQL query to determine the compatibility of map visualizations | 105 |
| 18. | SPARQL query generated to obtain the 5 most common values for a class and property | 114 |
| 19. | SPARQL query generated to obtain results | 114 |
| 20. | SPARQL query that retrieves at most the 5 most common classes instantiated by the values of a facet | 122 |
| 21. | SPARQL query that computes the most specific common superclass | 123 |
| 22. | Generated SPARQL query before pivoting | 124 |
| 23. | Generated SPARQL query after pivoting | 124 |
| 24. | SPARQL query after sorting by date and paginating | 150 |

Part I

Prelude

1.1. Motivation

It has been a long time since Tim Berners-Lee invented the World Wide Web (WWW) [BICP92] and meanwhile the Web has evolved and has become more sophisticated. Its huge success and its transition to the Semantic Web presents new requirements. Nowadays, the factors to consider when creating a website are not only technological but also of information structure, contents and functionalities.

Despite the fact that the Semantic Web was proposed more than ten years ago [BLHL01], it hasn't been until recently when it has started to become popular, especially thanks to Linked Open Data initiatives like those conducted by the USA, UK or Spanish government agencies towards a greater level of transparency. The objective of this initiative is to motivate the publication of Open Data in formats that are more easily integrable, queryable and that facilitate its reuse. As a result, the amount of data available in the Web, in its transition to a Web of Data or Semantic Web, is increasing at an astounding rate. The cloud of interrelated and open datasets included in the LOD cloud has rapidly evolved, from the 2 billion statements and 30 datasets one year after its creation in February 2007, to more than 31 billion statements¹ and 295 datasets in September 2012.

Visualizing and interacting with Linked Data is an issue that has been recognized from the beginning of the Semantic Web (cf. e.g. [GC02]). However, the Semantic Web has not been adopted by end users [SBLH06] yet. This is due in part to the fact that users find it difficult to use. Sometimes even advanced users of the Semantic Web find it complicated [HDS06]. The potential of this vast amount of data is enormous but in many cases it results in the need for handling the Information Overload phenomenon [Tho07]. It is very difficult and cumbersome for users to visualize, explore and use this data, especially for lay-users without experience with Semantic Web technologies. From the end-user perspective, the available datasets are monolithic and opaque files, which usually can just be explored using complex semantic queries or complex user interfaces. Interacting with Semantic Web data is hindered by users' lack of knowledge of semantic technologies, with common questions such as "what is a URI?", "what is RDF?", "what is an ontology?". The underlying structures in RDF as well as complex query languages like SPARQL are generally unknown to the user.

¹<http://www4.wiwiss.fu-berlin.de/lodcloud/state/>

Existing tools make it difficult for users to explore a dataset, most of them require technical skills and in many cases the results are not very usable. The transition of the Web to the Web of Data requires the support for a richer user interaction. While this growth of semantic data can generate a lot of opportunities, it requires new ways of accessing information. The richer knowledge representation models in the Semantic Web can be exploited to support new approaches for browsing, visualizing and searching the Semantic Web.

The objective is now to try to make all this data more usable, so users that are not Semantic Web experts, when facing a dataset, can easily grasp what kind of entities are contained therein, how they are interrelated, what are the main properties and values, etc. It is necessary to explore effective ways for displaying, browsing and querying this data. This will increase the awareness of the semantic data currently available in the Web and also facilitate the development of new and innovative applications on top of it. The overall outcome will be that available data increases its impact and the society as a whole benefits more from semantic data.

1.2. Problem statement

1.2.1. Defining End Users

One of the requirements of the Semantic Web is to be usable by both tech-savvy users and lay-users. The complexity of Semantic Web data limits its use to those who can read and understand how RDF and other technologies like SPARQL work [HRH08, Hea08b].

Different types of users can have different requirements in the use of the Semantic Web. Therefore, it is necessary to identify the main target user groups expected to use the Web of Data. Different user profiles have different skills, requirements and they carry out different tasks [SP04]. In this work we distinguish between three types of users:

- **Tech-users:** users with experience in software but also in Semantic Web technologies, who can understand RDF as a data format and are able to interpret an ontological model.
- **Lay-users:** users who may have knowledge about information technology but do not know about Semantic Web technologies, RDF or ontology models. This kind of users are able to find information in Internet through resources such as search engines or the Wikipedia. These users might be interested in using Semantic Web data to find particular data they are interested in.
- **Domain-expert users:** this kind of users may not necessarily have knowledge of software technologies but have an expert knowledge of a concrete domain. They are likely to have a good understanding of the data structure and contents, which allows them to interact with large amounts of complex and heterogeneous data. These users might be interested in using Semantic Web data for advanced domain-specific queries.

1.2. PROBLEM STATEMENT



| | |
|---|---|
|  | <p>Michael Harper is a 30 years old freelance developer who creates and commercialises mobile applications using online application stores. He is currently developing a mobile application that supports bird watching and as a way to reduce development costs to a minimum he is trying to reuse as much as possible available data about bird species, habitats, etc.</p> |
| <p>Christina Warren is a 23 years old journalist who is currently in charge of the Films section of an online journal. She likes to write about curious facts like “who appears most in films where Woody Allen is both the director and an actor”. However, these kinds of things are really difficult to find out using resources like Wikipedia or IMDb.</p> | <p>flickr.com/photos/electricnerve</p>  |

Table 1.1: Personas illustrating the intended users

Using the Personas approach [Gar10], we can illustrate the target audience as shown in Table 1.1. In the first scenario, Michael Harper can be considered a lay-user. In other contexts he could be considered a tech-user because he works as a software developer and knows about information technology. However, in our case, the most important characteristic to define user profiles is the knowledge of Semantic Web technologies. Therefore, in our context he is considered a lay-user.

Michael Harper could use the DBpedia [ABK⁺07] to find the information about bird species he is looking for. However, the DBpedia² is a large dataset and describes 3.77 million things, including 202.000 species. Its ontology covers 359 classes described by 1.775 different properties.

In the second scenario, Christina Warren can be considered a domain-expert. LinkedMDB³ contains information about the domain she is interested in. However, this dataset does not have an ontology, which makes it even more difficult to identify the main classes and properties. For the purposes of this work we consider domain-experts to be lay-users of the Semantic Web.

Both scenarios illustrate users who are interested in using Semantic Web data but do not have knowledge about Semantic Web technologies. Existing tools make it very difficult for non-technical users to explore a dataset, realize what kind of resources there are, what properties they have and how they are related.

²DBpedia 3.8, <http://www.dbpedia.org/>

³<http://linkedmdb.org/>

1.2.2. Human-Semantic Web Interaction

Despite the Semantic Web is designed for machine consumption, at the end, humans are its real consumers. However, lay-users face different barriers when diving into the Semantic Web [MMP⁺08]. For such interaction to occur, end users should have usable tools and simple methods to explore the Web of Data. The size and underlying technologies of the Web of Data presents several challenges for browsing and interacting with it:

Challenge 1: Exploration starting point Most of existing Semantic Web browsers assume the end user will start browsing from a specific URI. However, most of end users do not even know what a URI means. They need an exploration starting point [DRP11, DR11].

Challenge 2: Getting an overview As data set size increases, human ability to obtain a good mental overview and retain information in memory decreases. This poses a challenge for large amounts of complex and heterogeneous data [HMM00, HHG09, DR11]. In these cases, obtaining a good mental model of the data can be also a difficult task even for technical users and domain experts [SP04, MG03].

Challenge 3: Combating information overload Presenting all properties and relations of a given resource can lead to information overload [Tho07]. Most tools only provide access to single (but detailed) resources sequentially, which is very slow and easily saturates the user's working memory [ABDM04, DR11]. All this information should be presented in a more understandable form to lower the cognitive load [CS96].

Challenge 4: Advanced exploration An exploratory technique is necessary to explore large datasets [DKS07, ODD06]. Lay-users should be able to complex perform queries without having knowledge of Semantic Web technologies.

Challenge 5: Context information Semantic Web browsers should provide mechanisms to allow the navigation among resources while also providing context information to guide users to their target [SM10]. Users should be able to know what are they seeing, where they come from and where they can go. Otherwise, the quantity of data can make users feel lost after following several links between resources. That phenomenon was described in earlier days of the web as "Lost in Hyperspace" [EH88].

Challenge 6: RDF visual representation RDF [?] is the standard for resource semantic descriptions but mainstream users do not understand it. Unlike web pages, which are designed for human consumption, RDF is devoid of any presentation and only contains data. The results should be presented in a usable and understandable way for non-experts, hiding the complexity of the underlying technologies [DRP11, DHDZ10, HZ08, DR11].

1.3. HYPOTHESIS

Challenge 7: User interaction and navigation Users are familiar with the traditional Web and its browsable nature. In the traditional Web, browsers use links to navigate between web pages, while in the Semantic Web they use links to navigate between resources [Hea08b]. The user interaction should be replicated and adapted to the Semantic Web and Linked Data [DRP11].

Challenge 8: Scalability When facing the Web of Data, scalability becomes very important [SM10]. Applications based on Semantic Web data should be able to handle large datasets. In order to provide an acceptable user experience, user interfaces must not take too long to answer queries or freeze [NL06].

Challenge 9: Schema independence In some cases, Semantic Web datasets have no immediate access [ROH05] and they are only accessible through user interfaces for specific domains. Semantic Web applications should not be tailored to a specific schema. Schema flexibility is one of the main features of the Semantic Web and user interfaces should be capable of adapting and providing the best possible user experience independently of the specific data and schemas [DR11].

1.3. Hypothesis

The challenges presented above show that there are still many barriers in bringing the benefit of Semantic Web data to users. They difficult the user experience when interacting with semantic data, especially for users without knowledge in Semantic Web technologies. These challenges should be identified and considered when developing Semantic Web tools and evaluating their quality. Reducing the interaction barriers that these challenges might pose, and profiting from the new features that the Semantic Web offers, should improve users perception about tools based on Semantic Web technologies.

If non-technical end users are to use the Web of Data, they must employ tools that allow them to do so, focusing on the user interface (UI) and usability. User interfaces are a key part of supporting information discovery and exploration. By providing interfaces to the Semantic Web and making Semantic Web data more usable and accessible should demonstrate its utility and expand its uptake. Reusing interaction patterns and components from the traditional Web might facilitate the interaction and reduce learnability problems.

These work has been carried out with three hypotheses regarding the interaction with Semantic Web data:

Hypothesis 1 *The Visual Information-Seeking Mantra proposed by Shneiderman [Shn96] can be applied to design a user interface for interacting with Semantic Web data. The tasks described in the Mantra, "Overview first, zoom and filter, then details-on-demand", should be followed to provide users an effective data exploration.*

Hypothesis 2 *The challenges presented above can be addressed, with acceptable results, through an automatic Information Architecture [RM02], not designed for a specific and fixed schema. Thanks to Semantic Web technologies, the process of creating the Information Architecture of a website can be automated and adapted to different schemas. Moreover, as sometimes these schemas are incomplete or semantic data does not fully tally to them, the user interface should also take into account the actual structure of data.*

Hypothesis 3 *The previous automatic Information Architecture can be implemented using existing interaction patterns and components users are already familiar with. This facilitates the adoption by lay-users and improves learnability. Users are able to explore Semantic Web data without perceiving differences from the interaction with traditional websites.*

Hypothesis 4 *Through automatic Information Architecture components, users are capable of building complex queries while they interact with data without requiring knowledge about the underlying technologies or the vocabularies and specific terms used to structure the dataset.*

1.4. Contributions

The work in this thesis is cross disciplinary involving Human-Computer Interaction and Semantic Web. It presents the following contributions to the current state of Semantic Web research:

The first contribution is an overview of relevant research in the field of Human-Semantic Web Interaction, more specifically user interfaces for the exploration of Semantic Web data. We derive a set of challenges for interacting with Semantic Web data and we present a survey and classification of existing applications that have also tried to address these challenges.

The second contribution is the adoption of the tasks for data analysis proposed by Shneiderman [Shn96] for interacting with Semantic Web data. We provide a full analysis of these tasks applied to the context of Semantic Web exploration.

The third contribution is a set of algorithms and Information Architecture components to perform these tasks, which allow to explore Semantic Web data at different levels:

- An algorithm to generate an overview of datasets and the implementation of different overview components: navigation menus, site map, site index and treemap. These components allow users to get an idea about the overall structure of the dataset and provide a starting point for their navigation.
- A faceted browser with an automatic method to discover and rank those facets that best represent a dataset and allow efficient navigation. This method incorporates existing heuristics but also considers the descriptive value of properties. Our faceted browser also includes a pivoting functionality, which allows users to refocus their search to related items and create complex queries.
- A formal model to visualize Linked Data, which allows to dynamically connect data with visualizations. The unified RDF data model being prevalent on the Semantic Web enables us to bind data to visualizations in a dynamic way.

1.5. OUTLINE

We provide a comprehensive, generic and scalable implementation of these components, which allow users to explore semantic datasets and improve the usability and accessibility of Semantic Web data. These components have been evaluated with end users as part of a User Centred Design development process.

Our fourth and final contribution is a Semantic Web Exploration Tools Quality in Use Model (SWET-QUM), which allows to evaluate the quality of applications based on Semantic Web technologies. In our case, this model has guided the development and evaluation of our proposal. Moreover, it has also facilitated the comparability with other tools.

1.5. Outline

The complete outline of this document is as follows. The second part of this work provides the necessary background to contextualize this work. Chapter 2 presents the state of the art: the Semantic Web, its core technologies and the Linked Open Data initiative. A survey of existing tools and related work is presented in Chapter 3.

In the preparatory part we describe our approach and the methodology followed in this work. They guide the research work that has been done in the contribution part. Chapter 4 introduces our approach based on the Information Architecture domain and Shneiderman's tasks for data analysis. We explore these tasks in the context of interacting with Semantic Web data and we describe the Information Architecture components used to implement them. Chapter 5 details the methodology followed during the development of this work and proposes a model to evaluate the quality in use of tools to explore Semantic Web data.

In the contribution part of the thesis we focus on algorithms and components that solve the challenges identified. Chapter 6 proposes methods and algorithms necessary to explore Semantic Web data at different levels of detail and used to implement the Information Architecture components. Chapter 7 presents the iterative user interface development, based on the Information Architecture components identified. This chapter is divided into five different iterations.

This document concludes with conclusions and future work, presented in Chapter 8. We summarize the main contributions of this thesis and point out some open problems and future research directions.

Finally, Appendix A includes the documents used in the user tests: confidentiality document, post-task questionnaire and post-test questionnaire.

Part II

Background

Standards and Technologies of the Semantic Web

2.1. The Semantic Web

Tim Berners-Lee's vision of a Semantic Web is almost as old as the web itself. However, it took a few more years to be defined, starting 1999, when he wrote his book "Weaving the Web" [BLF99], where he described his dream for the Web and introduced the *Semantic Web*:

I have a dream for the Web... and it has two parts.

In the first part, the Web becomes a much more powerful means for collaboration between people. I have always imagined the information space as something to which everyone has immediate and intuitive access, and not just to browse, but to create [...]

In the second part of the dream, collaborations extend to computers. Machines become capable of analyzing all the data on the Web - the content, links, and transactions between people and computers. A "Semantic Web", which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy, and our daily lives will be handled by machines talking to machines, leaving humans to provide the inspiration and intuition [...]

Some years later, in 2001, Tim Berners-Lee described in more detail the Semantic Web [BLHL01] as an extension of the current Web, with contents aimed not only for humans but also for computer agents. The word semantic itself implies meaning or understanding. Meanwhile, the W3C published a set of standards, markup languages and official recommendations related with the Semantic Web. They are summarized in Figure 2.1.

Nowadays, the Semantic Web is a collaborative movement led by the World Wide Web Consortium (W3C). According to the W3C [W3C], "*The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries*".

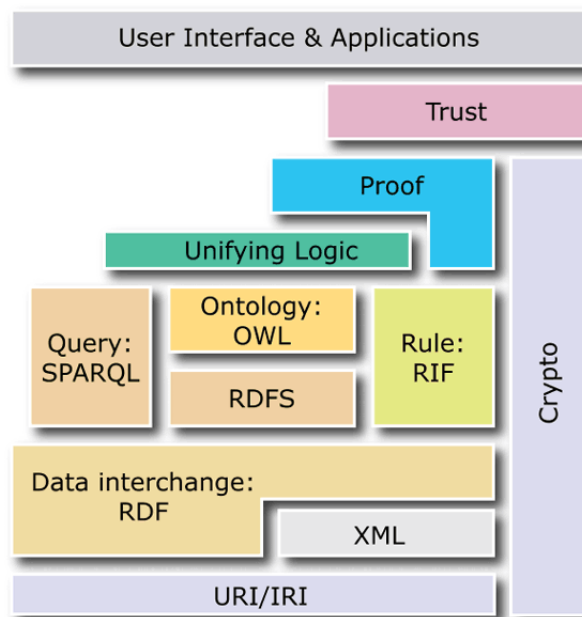


Figure 2.1: The Semantic Web layer cake, from [http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#\(24\)](http://www.w3.org/2007/Talks/0130-sb-W3CTechSemWeb/#(24))

2.1.1. From a Web of documents to a Web of data

One of the major problems of the current web is that the HyperText Markup Language (HTML) was originally designed to create and structure web resources. Web pages typically contain HTML markup that specifies links to related resources and tells web browsers how to display information. Computers are able to interpret such markup and display the structure of a web page, but its content and meaning, represented in natural languages, are only accessible to humans.

Most information on the web is designed for human consumption and machines can not easily understand its meaning. Humans are able to read and understand the meaning of a text or an image, but for a machine, HTML describes only a structure and does not have any semantics. Moreover, hyperlinks in HTML neither have any meaning. They indicate that a document is related to another, but not what is the relationship between them.

The aim of the Semantic Web is to merge web resources with machine-understandable metadata, better enabling people and computers work in cooperation. While the World Wide Web is a web of documents, the Semantic Web is a web of resources and metadata [Hea08b]. Data is no longer only expressed in natural languages, but also in formats that can be interpreted by machines. In the Semantic Web, information is given well-defined meaning.

2.1. THE SEMANTIC WEB

2.1.2. RDF

The Resource Description Framework (RDF) [Bec04] is a standard model for data interchange on the Web. It is the W3C standard for identifying resources and expressing statements and about them. It allows data to be mixed, exposed and shared across different applications. Using this model, data cannot only be viewed by humans, but also consumed and processed by applications.

RDF is based upon the idea of making statements about resources. Statements are stored as subject-predicate-object expressions, which are known as triples. In each triple, the subject denotes a resource and it is usually represented by a URI, which makes it globally identifiable. The predicate denotes an aspect of the resource and expresses a relationship between the subject and the object. The object is either a literal or a resource, also identified by a URI. The linking statements structure forms a labeled directed graph, composed by nodes and directed edges between nodes. The edges represent the named link (the predicate or property) between two resources, represented by graph nodes.

Figure 2.2, taken from the W3C website, describes “Eric Miller”: “there is a Person identified by <http://www.w3.org/People/EM/contact#me>, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr.”.

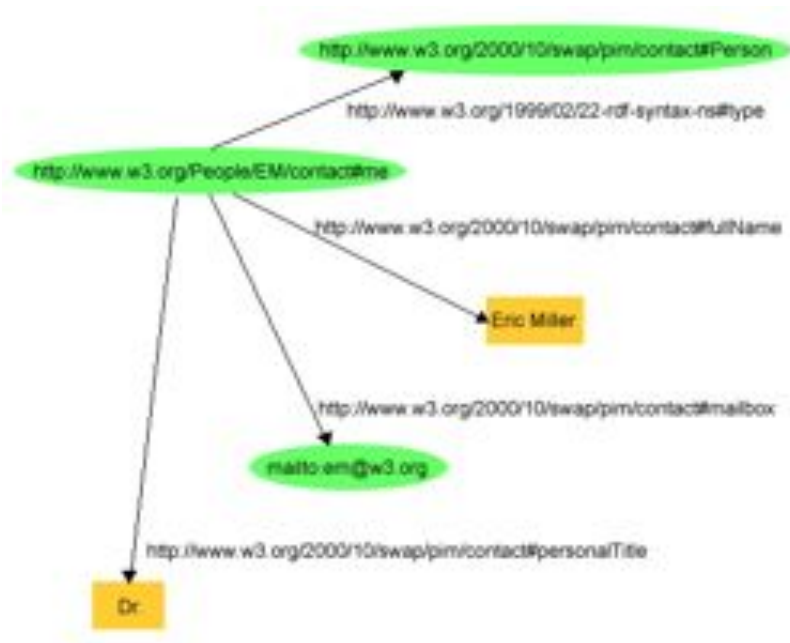


Figure 2.2: RDF graph describing Dr. Eric Miller, from <http://www.w3.org/TR/2004/REC-rdf-primer-20040210/>

The graph representation is often used to visually understand RDF. Apart from this representation, RDF can be serialized in different formats such as *N3*, *Turtle* or in *RDF/XML* [Bec04], as shown in Example 1.

```
1 <?xml version="1.0"?>
2 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3     xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">
4
5     <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
6         <contact:fullName>Eric Miller</contact:fullName>
7         <contact:mailbox rdf:resource="mailto:em@w3.org"/>
8         <contact:personalTitle>Dr.</contact:personalTitle>
9     </contact:Person>
10
11 </rdf:RDF>
```

Example 1: (RDF/XML Syntax)

2.1.3. RDF Schema

RDF Schema (RDFS) [GB04] is a semantic extension of RDF designed to create RDF vocabularies. It provides the basic elements to describe ontologies (more details in Section 2.1.4. These are the more relevant primitives of RDF and RDFS:

- `rdfs:Resource`: is the class of everything. All things described by RDF are instances of the class `rdfs:Resource` and all other classes are subclasses of this class.
- `rdfs:Class`: declares a resource as a class. A class models a concept with some characteristics. The definition of `rdfs:Class` is recursive: `rdfs:Class` is an instance of `rdfs:Class`.
- `rdfs:Literal`: is the class of literal values such as strings and integers. Literals can be plain or typed. `rdfs:Literal` is an instance of `rdfs:Class` and a subclass of `rdfs:Resource`.
- `rdfs:Datatype`: is the class of datatypes. All instances of `rdfs:Datatype` are a subclass of `rdfs:Literal`.
- `rdf:Property`: is the class of RDF properties. `rdf:Property` is an instance of `rdfs:Class`.
- `rdf:type`: is an instance of `rdf:Property` used to state that a resource is an instance of a class.
- `rdfs:label` is an instance of `rdf:Property` that can be used to provide a human-readable name of a resource.
- `rdfs:subClassOf` is an instance of `rdf:Property`. The triple `C1 rdfs:subClassOf C2` states that the class `C1` is a subclass of class `C2`. The `rdfs:subClassOf` property is transitive.
- `rdfs:subPropertyOf` is an instance of `rdf:Property`. The triple `P1 rdfs:subPropertyOf P2` states that the property `P1` is a subproperty of property `P2`. The `rdfs:subPropertyOf` property is transitive.
- `rdfs:range`: is an instance of `rdf:Property`. It is used to state that the values of a property are instances of one or more classes.
- `rdfs:domain` is an instance of `rdf:Property`. It is used to state that any resource that has a given property is an instance of one or more classes.

2.1. THE SEMANTIC WEB

By using these elements it is possible to define classes, hierarchies of classes, properties and hierarchies of properties.

2.1.4. Ontologies and OWL

In computer science, ontologies represent knowledge of a shared conceptualization [Gru93]. An ontology captures and formalises objects or concepts within a domain, their properties and relationships among those concepts.

Ontologies play a key role in the Semantic Web. The Web Ontology Language (OWL) [SD04] is a semantic markup language for authoring and sharing ontologies on the Web. OWL is developed as a vocabulary extension of RDF. In addition to modeling classes and instances, OWL allows the definition of class intersections and unions, object cardinalities and equivalence of concepts. Furthermore, properties can be transitive, functional, symmetric or inverse.

OWL is based on *Description Logics (DL)*, making it possible to reason about the entities within that domain. Software agents can infer and generate new knowledge.

2.1.5. XML and XML Schema

The Extensible Markup Language (XML) [Gro08] is a markup language to encode documents so information can be more easily shared. The design goals of XML emphasize simplicity, generality and usability over the Internet applications. Many applications have been developed to process XML data since it allows freedom in structure.

XML Schema describes the structure of XML documents. It can be used to define rules and restrictions to which an XML document must conform in order to be considered valid according to that schema. XML Schema provides simple and complex data types such as dates, numbers, strings, etc.

2.1.6. SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) [PS06] is an RDF query language to retrieve and manipulate data stored in RDF. It is considered as one of the key technologies of the Semantic Web and it has become an official W3C recommendation.

Example 2 shows an example of a SPARQL query, which comprises, in this order:

1. **Prefix declarations:** for abbreviating URIs.
2. **Dataset definition:** stating what RDF graph(s) are being queried
3. **A result clause:** identifying what information to return from the query
4. **The query pattern:** specifying what to query for in the underlying dataset
5. **Query modifiers:** ordering, slicing or rearranging query results

```
1 # prefix declarations
2 PREFIX dc: <http://purl.org/dc/elements/1.1/>
3 ...
4
5 # dataset definition
6 FROM <dataset.rdf>
7
8 # result clause
9 SELECT ?title
10
11 # query pattern
12 WHERE {
13   ?uri dc:title ?title .
14 }
15
16 # query modifiers
17 ORDER BY ?title
18 LIMIT 10
```

Example 2: Structure of a SPARQL query

The first line of the query defines a PREFIX for the Dublin Core namespace. In this way, it is not necessary to type the full URI each time it is referenced and it can be shortened with `dc:.` The SELECT clause specifies what the query should return, in this case, a variable named `?title`. SPARQL variables are prefixed with `?`. The FROM clause defines which graphs to use, in this case, pointing to a local file. The WHERE clause consists of a series of triple patterns. A graph pattern in a WHERE clause consists of a subject, predicate and object triple. The query matches the triple patterns in the WHERE clause against the triples in the RDF graph.

In this example, the triple in the WHERE clause matches any node with the `dc:title` property and binds its value to the variable named `?title`. The variable `?uri` is used to match any subject URI, but it is not returned in the result set because it is not stated in the SELECT clause. Finally, because there may be many possible results, the ORDER modifier is used to sort results and LIMIT 10 is specified to reduce the number of results.

SPARQL can be used to create more complex queries, as shown in Example 3. In this example, the first triple in the WHERE clause matches any node with the `vcard:FN` property and binds its value to the variable named `?name`. The OPTIONAL clause indicates a pattern that is not mandatory. If the group matches, the result is extended with the desired variables. Otherwise, the original result is returned. Example 4 shows an example of the results returned after executing the SPARQL query, including the variable `?age` when it is available.

```
1 PREFIX info: <http://somewhere/peopleInfo#>
2 PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
3
4 SELECT ?name ?age
5 WHERE
6 {
7   ?person vcard:FN ?name .
8   OPTIONAL {
9     ?person info:age ?age .
10  }
11 }
```

Example 3: SPARQL query with OPTIONAL clause, from <http://jena.apache.org/tutorials/sparql.html>

2.2. LINKED DATA

```
1 -----
2 | name           | age |
3 -----
4 | "Becky Smith" |     |
5 | "Sarah Jones" | 23  |
6 | "John Smith"  | 25  |
7 | "Matt Jones"  |     |
8 -----
```

Example 4: SPARQL query results

Another useful SPARQL operator is `FILTER`, which can be used to filter the results based on certain conditions. Example 5 shows an example of a SPARQL query using the `FILTER` clause to restrict the value of two variables. The first `FILTER` clause is used to find only those resources with the variable `?age` greater than 24. The second clause filters the variable `?name` based on a regular expression, similar to the SQL `LIKE` operator. As a result, this query retrieves the names and ages of people who are older than 24 and whose name contains the word "Smith".

```
1 PREFIX info: <http://somewhere/peopleInfo#>
2 PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
3
4 SELECT ?name ?age
5 WHERE
6 {
7     ?person vcard:FN ?name .
8     ?person info:age ?age .
9     FILTER (?age > 24) .
10    FILTER regex(?name, "Smith") .
11 }
```

Example 5: SPARQL query with `FILTER` clause, from <http://jena.apache.org/tutorials/sparql.html>

More details about SPARQL and more examples can be found at <http://www.w3.org/TR/rdf-sparql-query/>.

2.2. Linked Data

Linked Data [BHBL09] is a W3C movement about using the Web to connect datasets. Linked Data describes methods and a set of best practices for publishing and connecting structured data on the Web. It can be viewed as a subset of the Semantic Web movement.

Linked Data is built upon two standard Web technologies: HTTP and URIs. Entities are identified by URIs and they can be looked up simply by dereferencing the URI over the HTTP protocol. The HTTP protocol provides a mechanism for retrieving resources. URIs and HTTP are supplemented by RDF, which provides a generic data model to describe resources.

Linked Data is based on four principles [BL06]:

- Use URIs as names for identifying things.
- Use HTTP for URIs so that people can look up those names and retrieve information.
- When someone looks up a URI, provide useful information, using the standards, i.e. RDF and SPARQL.

- Include links to other URIs so that users can discover more things.

Among others, in recent years there has been several studies and research on publishing Linked Data [BCH07], searching Linked Data [TDO07], consuming Linked Data [HSTS10] or browsing Linked Data [DR11].

2.3. Linked Open Data

Open data is the idea of making data freely available for everyone to use. Data is shared without restrictions from patents or copyright. Open data has gained popularity with the rise of the World Wide Web and with the launch of open data government initiatives.

The Linked Open Data (LOD) community project aims to extend the Web by publishing Open Data using Linked Data principles. Nowadays, the LOD has grown to 295 datasets, 31 billion RDF triples, interlinked by around 504 million RDF links. Figure 2.3 shows the LOD cloud. Each node in the diagram represents a distinct dataset published as Linked Data. The arcs indicate that RDF links exist between resources in the two connected datasets.

In 2010, Tim Berners-Lee proposed a 5-star rate scheme [BL06] to encourage users and organizations to expose their datasets as part of the Linked Open Data cloud:

- **1 Star:** “data is available on the Web (whatever format), but with an open license”.
- **2 Stars:** “data is available as machine-readable structured data (e.g., Microsoft Excel instead of a scanned image of a table)”.
- **3 Stars:** “data is available as (2) but in a non-proprietary format (e.g., CSV instead of Excel)”.
- **4 Stars:** “data is available according to all the above, plus the use of open standards from the W3C (RDF and SPARQL) to identify things, so that people can link to it”.
- **5 Stars:** “data is available according to all the above, plus outgoing links to other peoples data to provide context”.

In recent years a significant number of datasets have been published as Linked Data. In particular, Linked Open Data has become popular especially thanks to initiatives conducted by governments such as United Kingdom⁴ or USA⁵. They have adopted LOD for making distributed information publicly available [HSM⁺10].

⁴<http://data.gov.uk>

⁵<http://data.gov>

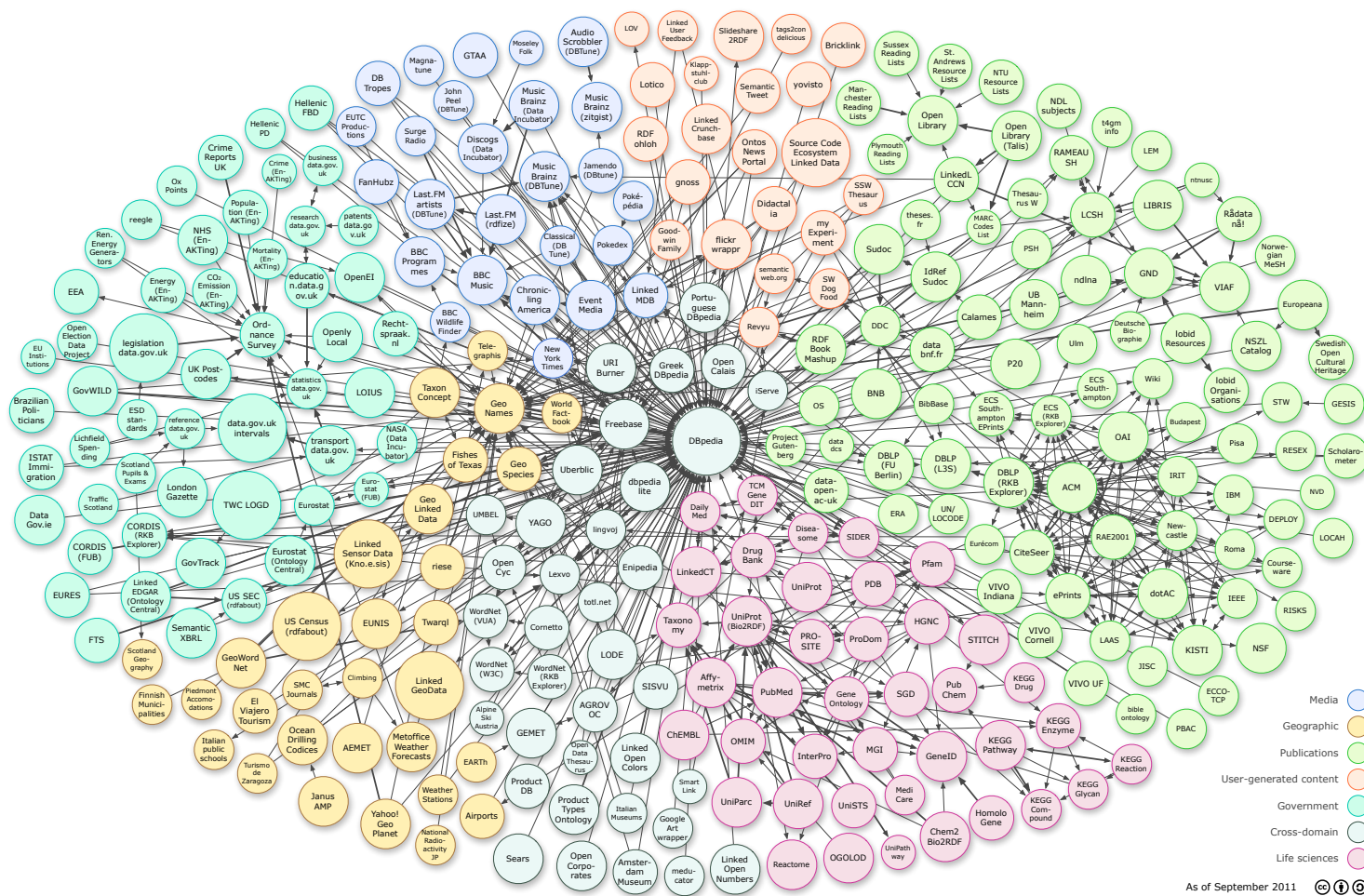


Figure 2.3: Linked Open Data (LOD) cloud, from <http://lod-cloud.net/>

Publishing and presenting Semantic Web Data in an accessible way for users has been addressed by several projects and different kind of tools. In this chapter we present existing initiatives that are related with this work. It is organized in different categories.

3.1. Semantic Web browsers

The first tool that comes to mind when trying to realise what a dataset is about is a Semantic Web browser. A number of tools for browsing the Semantic Web and for presenting RDF data have been developed so far. Semantic Web browsers differ from Web browsers because they are not prepared for navigating documents but triples, the fundamental building block of the Web of Linked Data.

3.1.1. Text-based browsers

Text-based browsers are general-purpose tools for presenting Semantic Web data. These tools render RDF data triples as HTML, using tables or lists to display properties, values and relationships. Some provide advanced methods for searching and filtering such as faceted browsing.

Disco

The *Disco - Hyperdata Browser* [BPGB07] is a simple browser for navigating Semantic Web resources. It renders as an HTML page all the information that can find about a specific resource. Users must provide the URI of a concrete resource to start navigation. Retrieved information is displayed as a property-value table. *Disco* also renders hyperlinks that allow users to navigate between related resources.

Tabulator

Tabulator [BICC⁺06, BIHL⁺] is a generic RDF browser and editor. In addition to the rendering of properties and values for a resource, *Tabulator* provides specialised visualisations like maps for geo-located resources or timelines for time-framed ones. Figure 3.1 shows an example dataset being browsed with *Tabulator*.

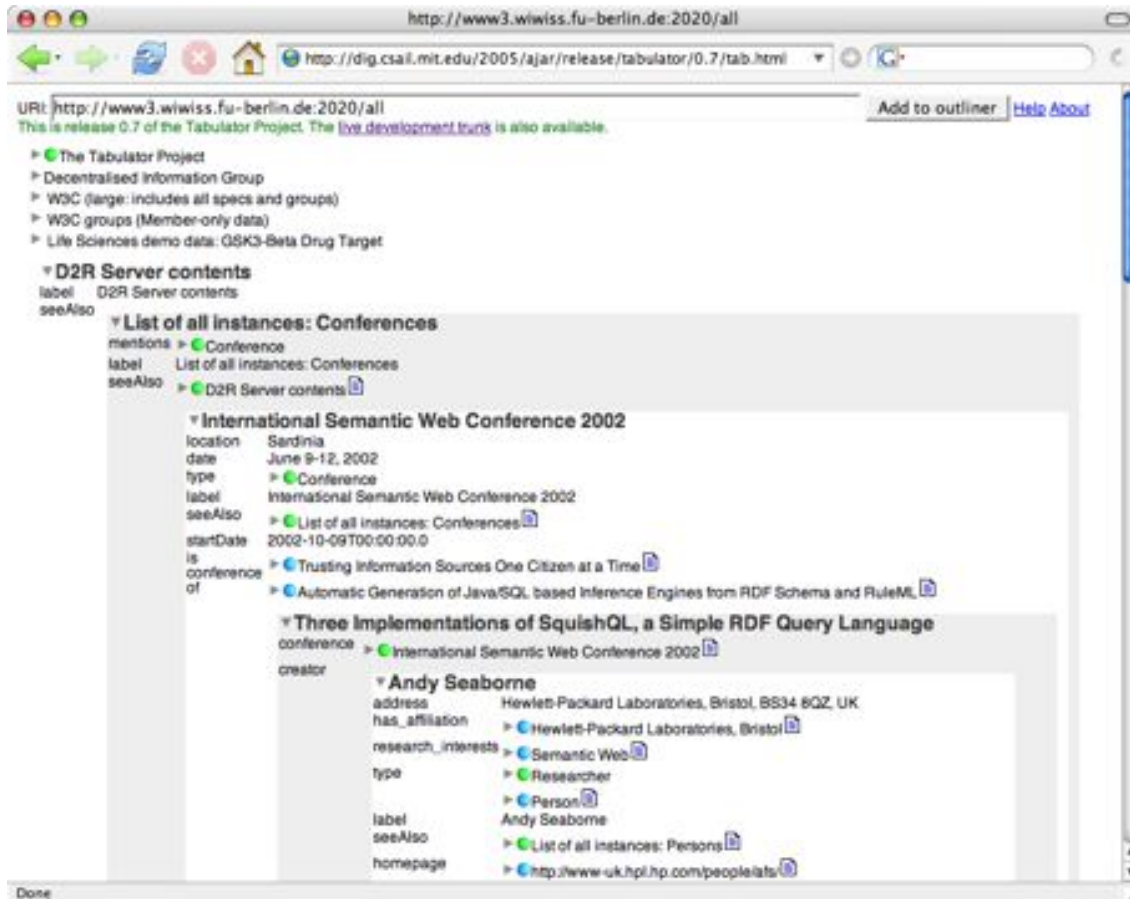


Figure 3.1: Tabulator RDF browser

Explorator

Explorator [dASB09] is a tool that makes it possible to browse a dataset available through a semantic queries service. Though *Explorator* makes it possible to browse the dataset by combining search, facets or operations on sets of resources, it makes it also difficult to get a broader view on the dataset other than a list of all the classes or properties used. Furthermore, its interface is difficult to understand for lay-users.

3.1. SEMANTIC WEB BROWSERS

Longwell

Longwell [lon] is a tool part of the *Simile Project*, which provides a graphical user interface for generic RDF data exploration in a web browser. It provides a faceted browser that allows users to search large models by filtering through properties and values. *Longwell* shows a list of the currently filtered resources (RDF subjects) in the main part of the screen and a list of filters in the side. Each filter corresponds to a property of the resources with its values and their frequency. It can be configured to choose and prioritize which facets should be shown when the page loads, or it can choose heuristically which are the most important and should be selected.

/facet

/facet [HvOH06] is a generic browser for heterogeneous Semantic Web repositories. Users can select and navigate facets of resources of any type. It provides also a time-related facet visualization and can display geographical information on yahoo maps.

Exhibit

Exhibit [HKM07] is a publishing framework for interactive web pages. It helps users to create interactive sites with advanced text searching and filtering functionalities. It also provides visualizations such as maps, timelines or charts. However, it requires a domain-expert to configure the different facets. Moreover, it can not work directly on SPARQL endpoints and the system does not scale well. Figure 3.2 shows the *Exhibit* interface browsing US presidents.

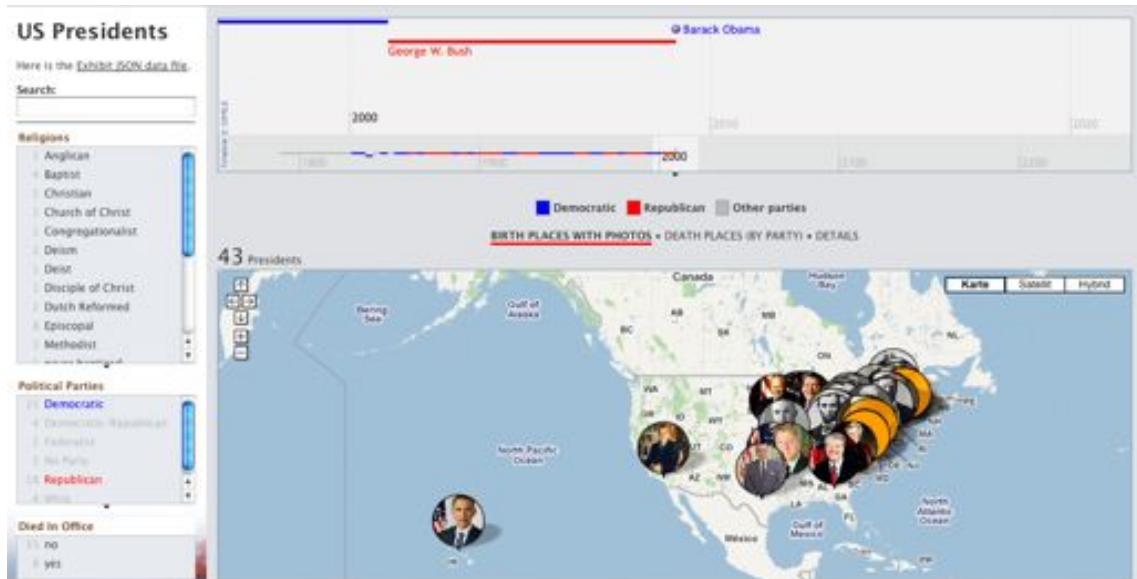


Figure 3.2: Exhibit interface

DBpedia Faceted Browser

The *DBpedia Faceted Browser* [HBS⁺10] is a project from Neofonie that allows users to make complex queries against DBpedia [ABK⁺07]. It supports keyword queries and offers relevant facets to filter search results, based on the DBpedia Ontology. The *DBpedia Faceted Browser* is a useful tool for browsing the DBpedia but it is not a generic browser, it only works for this concrete dataset. Figure 3.3 shows a screenshot of this tool.

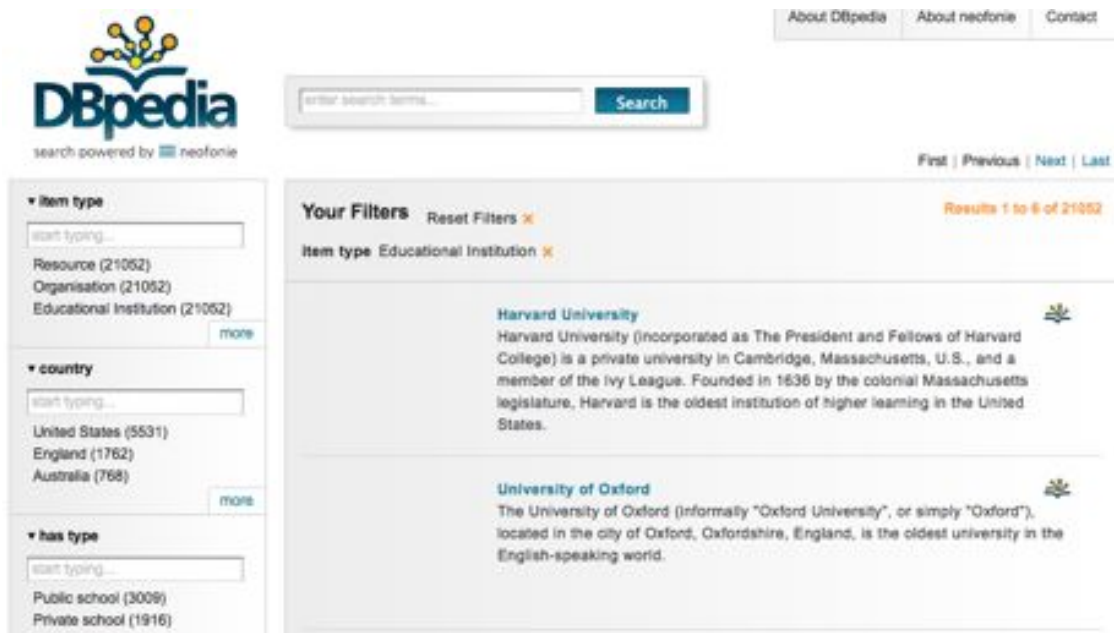


Figure 3.3: DBpedia Faceted Browser

Marbles

Marbles [mar] is a text-based RDF browser that retrieves information about resources by querying Semantic Web indexes and search engines. It is necessary to provide a URI as input or it can be used also as a SPARQL endpoint. It formats resources for HTML clients using *Fresnel* lenses [PBKL06]. *Fresnel* is a vocabulary for rendering RDF resources as HTML.

BrowseRDF

BrowseRDF [ODD06] is a faceted interface for arbitrary RDF data. Users can browse a dataset by constraining one or several of the facets using different operators: basic selection, join selection, inverse selection, etc. The authors also propose three metrics to rank facets automatically and choose those that best navigate the dataset.

mSpace Faceted Browser

The *mSpace* project [msWRS06] is an interface for browsing Semantic Web data. In *mSpace*, facets are horizontal and directional [WAs08], i.e. the display order of the facets is significant. A selection in one facet only alters facets in a single direction. This occurs left-to-right. This approach requires fewer queries since only some facets change. However, facets on the left become useless since they are not modified with selected values. Moreover, this interface is not applicable for large datasets because a large number of facets would make it unusable. Figure 3.4 shows part of the *mSpace* interface.

Sig.ma

Sig.ma [TCC⁺10] is a browser that integrates semantic data from multiple sources. The user can begin its navigation with a free text search, which is more useful for lay-users than entering a URI. *Sig.ma* is built on top of *Sindice*, a semantic search engine [TDO07].

3.1. SEMANTIC WEB BROWSERS



Figure 3.4: Facet columns in mSpace Faceted Browser

3.1.2. Graph-based browsers

Another common used visualization approach is to show ontologies as a graph. Classes are represented as nodes and edges represent relations between classes. In the following, we describe three contributions that belong to this category.

RDF-gravity

RDF Gravity [rgr] is a tool for visualizing RDF/OWL ontologies. It provides a graph visualization including different node shapes and edge decorations to distinguish different resource types. The tool allows users to specify filters to have specific views on the graph. It is also possible to perform a text search and SPARQL queries over classes, properties and instances. Figure 3.5 shows a screenshot of this tool.

IsaViz

IsaViz [Pie06] is an interactive RDF graph browser and editor. It provides a 2.5D user interface that allows to zoom and navigate through the graph. *IsaViz* can render RDF graphs using *Graph Stylesheets (GSS)*, a stylesheet language derived from CSS and SVG for styling RDF models represented as node-link diagrams. It also supports the use of *Fresnel lenses* [PBKL06] to display resources of interest.

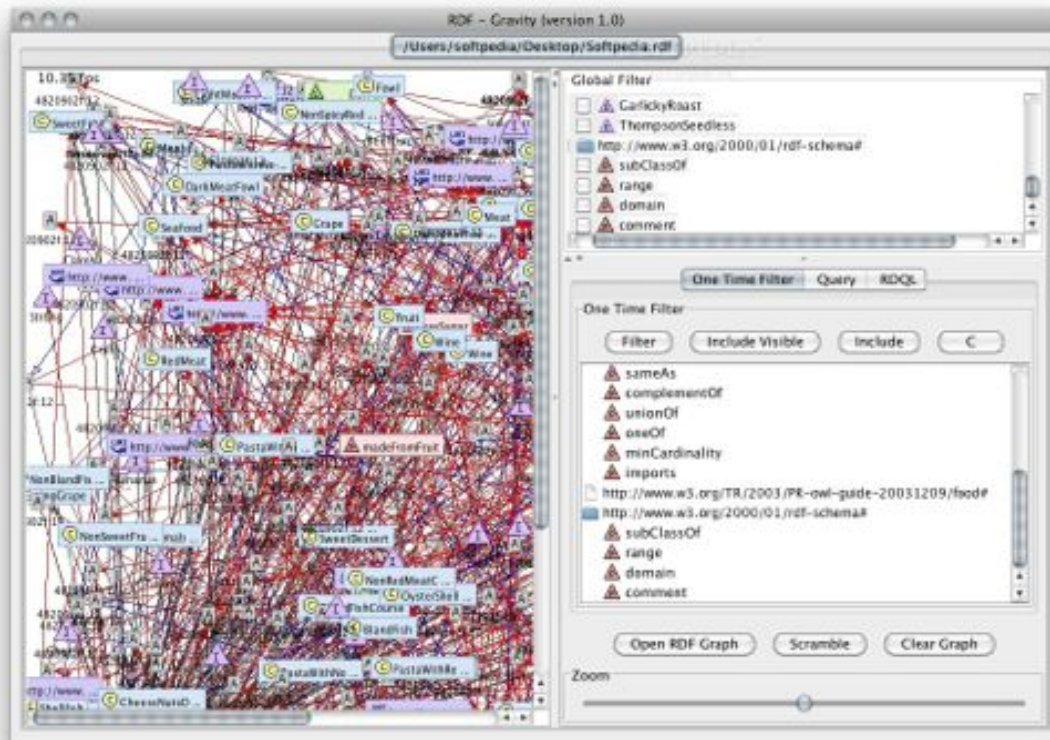


Figure 3.5: RDF-gravity interface

Fenfire

Fenfire [HCB08] is a generic RDF browser and editor. The user interface is a graph visualization of the RDF data model. To allow scalability to large amounts of data, it displays only a central node and its neighbours. However, it is necessary to provide a URI to start navigating and does not provide any overview support.

3.1.3. Browsers Supporting Pivoting

Beyond these single-focus browsers, there exist also multi-focus faceted browsers [CNF09] that provide a pivoting operation. In addition to letting a user filter results on simple properties, these browsers allow the user to “pivot” or refocus to properties of related objects. For example, a user querying for film directors born in Spain, can pivot or refocus to the set of movies directed by these directors. More details about multi-focus browsers and pivoting can be found in Iteration 2 (Section 7.2).

Virtuoso OpenLink Data Explorer

Also known as *OpenLink RDF Browser* [Erl], *Virtuoso OpenLink Data Explorer* is a web browser for interacting with Linked Data that provides a basic faceted view. It requires an entity URI as input or a text string to look for. Consequently, the facets view is limited to the resources retrieved from a previous search. Moreover there is no way to previously get an overview of the kinds of resources in the dataset.

3.1. SEMANTIC WEB BROWSERS

SParallax

Parallax [HK09] was one of the first browsers to offer pivoting but it was originally tied to *Freebase*, a large community database. Its derived tool, called *SParallax*, can work on top of SPARQL endpoints. *SParallax* shows the set of resources, accompanied by a list of facets for filtering. It also provides a list of connections, showing those properties that can be used in a pivoting operation. Though *SParallax* provides pivoting, it does not provide an overview feature. Moreover, its performance is limited and slow when exploring large-scale datasets.

Visor

Visor [PSHS11] is a generic RDF data explorer that can work over SPARQL endpoints. In *Visor*, exploration starts by selecting a class of interest from the ontology. Then, users can pivot to related collections and continue browsing. Despite it provides a hierarchical overview of the dataset, it is complicated to filter resources because there is not any faceted interface. Furthermore, its performance is very limited and does not allow the exploration of really large-scale datasets. Moreover, it is not clear whether or not it is a tool suited for lay users. Figure 3.6 shows a screenshot of the *Visor* interface.



Figure 3.6: Visor interface

gfacet

gFacet [HZL08] is a tool that provides pivoting based on a graphical representation. With *gFacet* it is possible to filter one class and then pivot to a related class keeping those filters for the instances of the second class connected to the filtered instances in the first class. However, the use of a graphical representation makes the user interface difficult to manage, especially for lay users not aware of the underlying graph data model. This is due to the fact, that boxes and links easily fill the screen and there is not a contextualisation that helps users understand what they are asking through the user interface, i.e. the underlying query that has been built through their interaction. Another shortcoming is that there is not an initial overview that helps users understand the shape of the dataset they are interacting with and where they can start from.

tFacet

tFacet [BH11] provides a functionality similar to pivoting. It is based on what the authors call hierarchical facets. However, these are really subfacets, i.e. facets of the entities accessible through a facet are shown in a hierarchical way under the superfacet. Unfortunately, this becomes impractical when traversing many different classes as the tree of facets becomes difficult to manage. Consequently, this tool cannot be considered suitable for lay users in its current state. Moreover, this approach constrains the kind of queries to be built more than pivoting. If the user starts from a class, the queries generated are about retrieving resources of that class that satisfy the filters for direct properties or properties of the classes related to it. It is not possible to switch to a different class and retrieve its instances as query result.

3.2. Ontology Visualization

In the Information Visualization domain we can find several useful techniques for visualization of hierarchically structured datasets [CMS99]. In concrete, Katifori et al [KHL⁺07] present a survey of ontology visualization techniques and categorize their characteristics. The techniques developed for this challenge can be mainly classified in *node-link* diagrams and *space-filling* visualizations. The traditional node-link representations such as *SpaceTree* [PGB02] or *Cone Trees* [RMC91] combine the conventional layout of trees with zooming interaction. However, they make poor use of the available display space and leave the root side of the tree completely empty. They do not give an effective overview of large hierarchies and nodes need to be expanded manually. When data gets larger the screen quickly becomes overcrowded with nodes and their children.

On the other hand, *space-filling* techniques such as *Treemaps* [Shn92] or *Crop Circles* [WP06] make a better use of the space by partitioning it into a collection of geometric shapes representing the tree structure. *Treemaps* use a rectangle to show the tree root and its children. Each child has a size proportional to the cumulative size of its descendants. They are a good method to display the size of each node in a hierarchy. In *CropCircles* each node in the tree is represented by a circle. Every child circle is nested inside its parent circle. The diameter of the circles is proportional to the size of their descendants.

Some evaluations and comparisons have been performed on these visualization systems. Kobsa evaluated six visualization techniques with hierarchical data to compare their efficacy and user satisfaction with them [Kob04]. Barlow et al. compared four different visualizations to evaluate the users' ability to understand the topology of the hierarchy and comparisons of node size [BN01]. Their results show that each technique has its strengths and weaknesses depending on the exploration tasks and dataset characteristics.

However, all these approaches focus on visualizing hierarchical structures and ontologies rather than exploring Semantic Web data. These studies have mainly focused on users performing tasks related with understanding the topology and navigating through the hierarchy. None of them has performed an analysis of whether or not these visualization systems can be used to perform real end-users tasks and if they provide good overviews of datasets.

3.3. CMS and Semantic Wikis

Other alternatives are Content Management Systems (CMS) or Wikis with semantic capabilities. Some mainstream CMSs and wiki systems have started to incorporate semantic technologies. The most significant case is the last version of *Drupal*⁶ that provides features such as semantic metadata storage, querying, importation or rendering using RDFa [CCPD09].

However, semantic CMSs, like *Drupal*, or portal building solutions, like *ODE-SeW* [CLCGP06], are intended more for content creation. They make it harder to publish an existing dataset and making its structure available to the user for exploration. It is usually necessary to build this structure manually using features like templates or content creation kits.

The same applies to semantic wikis. Semantic MediaWiki [KVV06] makes it possible to mix wiki mark-up with semantic annotations. OntoWiki [ADR06] provides support for distributed and collaborative authoring of RDF knowledge bases. However, semantic wikis are intended more for content creation than for importing and exploring existing data.

3.4. Summary

Dadzie and Rowe [DR11] present the most exhaustive and comprehensive survey to date of existing approaches to visualising and exploring Semantic Web data, particularly Linked Data. They conclude that most of the tools are designed only for tech-users and do not provide overviews on the data.

Semantic Web Browsers are especially useful when dealing with a dataset published as Linked Data because they provide a smooth browsing experience through the graph, e.g. *Disco* [BPG07] or *Tabulator* [BICC⁺06, BIHL⁺]. However, most of them do not provide additional support for getting a broader view of the dataset being browsed, just a view on the current resource. The presentation of the data is mostly limited to tabular listings of all resources, properties and values.

Other tools like *Explorator* [dASB09] or *Marbles* [mar] allow to browse a dataset available through a semantic queries service. In some cases it is also possible to get more informative components like facets, e.g. */facet* [HvOH06] or *BrowseRDF* [ODD06]. *Explorator* also makes it possible to browse the dataset by combining search, facets or operations on sets of resources. However, it is still difficult to get a broader view on the dataset other than a list of all the classes or properties used. Moreover, in some cases, facets are pre-computed and just available for a given dataset as in the case of the *DBPedia Faceted Browser* [HBS⁺10]. In other cases, tools begin to suffer from performance problems when dealing with large datasets, e.g. *Exhibit* [HKM07].

Some tools provide advanced filtering through pivoting. However, in some cases their functionality is limited to uni-directional links, e.g. *Sparallax* [HK09], or subfacets, e.g. *tFacet* [BH11]. In other cases, it is not clear whether their user interface is suitable for lay-users, e.g. *Virtuoso OpenLink Data Explorer* [Erl], *gFacet* [HZL08] or *Visor* [PSHS11].

⁶<https://drupal.org/>

Graph-based tools such as *Fenfire* [HCB08], *RDF-Gravity* [rgr] or *IsaViz* [Pie06] provide node-link visualizations of the datasets and the relationships between them. Although this approach can help obtaining a better understanding of the data structure, in some cases graph visualization does not scale well to large datasets [VFTjH05], as shown in Figure 3.5. Sometimes the result is a complex graph difficult to manage and understand [KS06].

Table 3.1 shows a summary of Semantic Web browsers. Our survey is not exhaustive. We have excluded projects that seem to be defunct at present, which makes it impossible to analyze them in more detail, e.g. *Humboldt* [KD08], *Haystack* [QK04] or *VisiNav* [Har10]. The table considers the following features: data overview, faceted browsing, pivoting, resource details, breadcrumbs, specific visualizations, generic or domain-specific and suitable for lay-users.

To summarize, most of existing tools make it difficult for non-technical users to explore Semantic Web data efficiently. They all provide details of concrete resources, but obtaining an overview cannot be easily done with most tools. There is little or no support to obtain overview information quickly and easily at the beginning of the exploration of a new dataset. Some provide facets and pivoting, but it is not clear whether they can be considered usable for lay users. Moreover, with few exceptions, published reports on these tools lack of formal usability studies with end users.

As a result, it is still difficult for many users to use most tools, to comprehend what kind of structures and resources are available, what properties resources typically have and how they are mostly related with each other. This can be a serious limitation when exploring Semantic Web data.

| Tool | Overview | Facets | Pivoting | Details | Breadcrumbs | Visualizations | Generic | Lay-users |
|-------------------------|----------|--------|----------|---------|-------------|----------------|---------|-----------|
| Disco | - | ✓ | - | ✓ | - | - | ✓ | ✓ |
| Tabulator | - | - | - | ✓ | - | Map, Calendar | ✓ | ✓ |
| Explorator | - | ✓ | - | ✓ | - | - | ✓ | - |
| Longwell | ✓ | ✓ | - | ✓ | ✓ | - | ✓ | ✓ |
| facet | - | ✓ | - | ✓ | ✓ | Map, Timeline | ✓ | ✓ |
| Exhibit | - | ✓ | - | ✓ | ✓ | Map, Timeline | ✓ | ✓ |
| Dbpedia Faceted Browser | ✓ | ✓ | - | ✓ | ✓ | - | - | ✓ |
| Marbles | - | - | - | ✓ | - | - | ✓ | ✓ |
| BrowseRDF | - | ✓ | - | ✓ | ✓ | - | ✓ | ✓ |
| mSpace | - | ✓ | - | ✓ | - | Map | ✓ | ✓ |
| Sig.ma | - | ✓ | - | ✓ | - | - | ✓ | ✓ |
| RDF-Gravity | - | - | - | ✓ | - | Graph | ✓ | - |
| IsaViz | - | - | - | ✓ | ✓ | Graph | ✓ | - |
| Fenfire | - | - | - | ✓ | - | Graph | ✓ | - |
| OpenLink Data Explorer | - | ✓ | ✓ | ✓ | - | - | ✓ | - |
| SParallax | - | ✓ | ✓ | ✓ | ✓ | Map | ✓ | ✓ |
| Visor | ✓ | ✓ | ✓ | ✓ | - | - | ✓ | - |
| gFacet | - | ✓ | ✓ | ✓ | - | - | ✓ | - |
| tFacet | - | ✓ | - | ✓ | - | - | ✓ | - |

Table 3.1: Comparison of Semantic Web data exploration tools

3.5. Rhizomer

The survey [DR11] presented in the previous section is used to situate our contribution, implemented in a tool called *Rhizomer* [GGP⁺08] and available online⁷. *Rhizomer* can be classified mainly in the category of text-based visualisation tools, though it also includes graphical representations for dataset overviews and visualizations. However, it is important to notice that it is not intended as a Linked Data browser. It is geared towards publishing a dataset and generating the user interface to improve user interaction for that specific dataset.

First of all, *Rhizomer* is based on a simple architecture, which makes it flexible, scalable and capable of adapting to different deployment and use scenarios. Its core is rooted on simple HTTP mechanisms and follows a REST approach [RR07]. *Rhizomer* also implements content negotiation taking into account the requested content type thus providing the requested data in the desired format.

Each resource is managed through the URI referencing where it is published, thus basing the whole system on a Resource Oriented Approach. The basic HTTP commands allow managing each resource: GET retrieves the semantic data associated with the resource in the requested format, PUT updates the data for the resource with the submitted one, POST creates a new resource with the submitted semantic description and DELETE removes the specified resource and the corresponding data.

All the previous HTTP commands are forwarded to the underlying data store, see Figure 3.7. Currently, *Rhizomer* integrates connectors for Jena and Virtuoso. These connectors make it possible to implement all the data management operations.

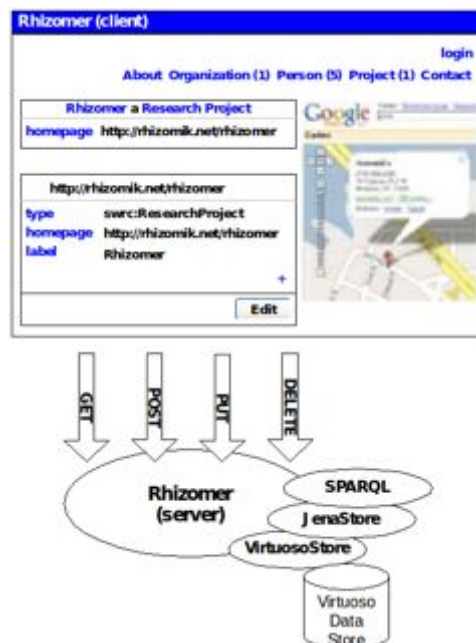


Figure 3.7: Rhizomer architecture overview

⁷<http://rhizomik.net/rhizomer/>

The client-side functionalities have been developed with the aim of improving the usability of the user interface. They are deployed in the user's browser and implemented using JavaScript and asynchronous HTTP calls (AJAX [CPJ05]), though the basic functionality is also available without JavaScript in order to improve accessibility [GGP⁺10].

Like many Semantic Web browsers and data publishing tools, *Rhizomer* provides an HTML view on the data that also facilitates the navigation across the data graph. The RDF syntax of semantic data is completely hidden in order to increase usability. However, as it has been shown in related tools, this approach does not contribute towards an awareness of the overall structure of a dataset. Neither does it provide ways to explore data and perform complex queries. Therefore, our goal is to improve *Rhizomer* and provide users more functionalities in order to explore Semantic Web data.

Part III

Preparation

To overcome human-Semantic Web interaction challenges, our proposal is to explore the tasks for data analysis proposed by Shneiderman [Shn96] and to draw from the experience in the Information Architecture domain [RM02], adapting them to the context of the Semantic Web. Our approach is intentionally simple, situated in the Web context. Our objective is that users can interact with Semantic Web data without perceiving any difference from traditional websites.

4.1. Information Architecture

Information Architecture (IA) is the art and science of organizing information. In the context of the World Wide Web, Information Architecture [RM02] is the discipline that organizes and labels the information on websites. This discipline englobes analyzing the contents, organizing web pages and designing the navigation systems.

The challenge of structuring and presenting semantic data to end users can be addressed with the experience accumulated in the Information Architecture domain. Information Architecture focuses its efforts in this problem, especially in complex systems and situations with large amounts of information. A good IA can improve the quality of a website and users can find easier the information they are looking for.

Traditionally, User-Centered Design techniques are used to develop the Information Architecture of websites. The typical one is Card Sorting [SG09], where users are given a set of cards labelled with the main topics of the site and they group these cards following their own criteria. In order to generate an IA as meaningful as possible for the broader range of users, the card sorting is repeated with different users. This technique requires a lot of time and effort from developers and most of this effort is wasted as soon as the content structure is established and fixed. Then, the Information Architecture becomes something static. If new kinds of items are introduced or a part of the content becomes more relevant, the Card Sorting should be repeated, at least partially.

In the case of web sites built on top of semantic data we have the opportunity to automate part of the process of generation and maintenance of the Information Architecture. The RDF data model enables the use of automatic tools to process it in a more sophisticated way. This is possible because semantic data is structured by thesauri and ontologies, which hierarchically organise the kinds of things described in the dataset. They specify all the classes or concepts but also which entities belong to a certain class or are related to a certain concept.

Information Architecture identifies four kinds of systems:

- **Organisation systems:** they present information in different ways, following different schemas that make it possible to group or differentiate information using different criteria, like chronological or alphabetic order.
- **Navigation systems:** they help users move across the available information. For instance, there are navigation bars or site maps.
- **Labelling systems:** they describe categories, options and links using terms that are meaningful for users. They are all around the information architecture of a site, even as part of other systems, e.g. navigation bars labels.
- **Search systems:** they allow users to search specific information based on some sort of keywords. They also offer mechanisms to restrict the search space.

The drawback of all these IA systems is that they are quite expensive to develop and maintain. Nowadays, when developing a website, the procedure usually begins by defining the Information Architecture for the domain of this site with the help of the future users of the website. The obtained Information Architecture is usually based on formalisms that allow to represent only a small part of the domain semantics. Therefore, the process of creating a website from this type of Information Architecture is a heavy process that requires a lot of time and effort by developers, mainly because little automation can be accomplished.

Fortunately, when these IA systems are built on top of the highly structured data typical in the Semantic Web and Linked Data, it is possible to automate most of the development and maintenance work. The Semantic Web provides methods and tools to model the information architecture of a concrete domain with more detail and in a formal way. This allows the use of automatic tools to process it in a more sophisticated way.

It is possible to establish a correspondence between these IA systems and the Semantic Web. Data on the Semantic Web is modelled and represented using ontologies and vocabularies, which allow to organize the information, i.e. organisation systems. Labelling systems can be automatically obtained from different classes and properties such as `rdfs:label`, SKOS concepts [MPA07] or subjects, etc. Finally, navigation systems and search systems such as navigation menus, site maps or faceted browser can be automatically generated thanks to the prevalent RDF data model.

4.2. Tasks for data analysis

As the volume of information available in the Web of Data increases, interacting with data becomes a more difficult task. To interact with large datasets, the literature proposes to divide the task of exploring a dataset into different stages. Shneiderman [Shn96] presents the Visual Information-Seeking Mantra: “Overview first, zoom and filter, then details-on-demand”. The Mantra describes the fundamental set of tasks for data analysis and how data should be presented to users in order to achieve an effective exploration. It is presented as a guideline for how to design visual interfaces, where the first stage they present should be to gain an overview of the entire collection.

Our starting point is the fundamental set of tasks for data analysis proposed by Schneiderman. We have explored the most appropriate Interaction Patterns [Tid05] to perform these tasks and the Information Architecture components to implement each pattern (Table 4.1):

| Task | Interaction Patterns | Information Architecture components |
|---------------|---|---|
| Overview | Global Navigation Directory Navigation | Navigation menus Site map Site index Treemap |
| Zoom & Filter | Exploratory search | Facets Paginating results Sorting results |
| Details | Details on Demand | HTML representation Specific visualizations |
| Relate | | Links to related resources Pivoting in facets |
| History | Breadcrumb navigation | Location breadcrumbs Path breadcrumbs Attribute breadcrumbs |
| Extract | | Bookmarks |

Table 4.1: Tasks for data analysis, interaction patterns and Information Architecture components

Our proposal is to elaborate the selected tasks, interaction patterns and components in the context of semantic data. We have chosen these patterns and components because they are simple and very common so users are very comfortable using them. They are part of the “culture” about how information is presented in the Web so they can be easily learned. The objective is to adapt existing IA components to provide this interaction to users. Although these components look like the common ones, they are capable of giving access to the richer semantic data they are built on top of.

In the following subsections we describe each task and the main components used to perform them.

4.2.1. Overview

Obtaining an overview is an important first step in the exploration of data. The objective is that the user is capable of getting an idea about the overall structure of the dataset. Greene et al. [GMPS00] argued that a good overview “provides users with an immediate appreciation for the size and extent of the collection of objects the overview represents, how objects in the collection relate to each other, and importantly, what kind of objects are not in the collection”. Overviews provide a general context for understanding the available information and can be used as starting point for navigation.

However, obtaining this overview can not be easily done with most of the existing Semantic Web browsers. Most of existing browsers assume the end user will start browsing from a specific URI, but most of end users do not even know what a URI means. They need an exploration starting point. Moreover, these browsers only provide access to single (but detailed) resources sequentially, which is very slow and easily saturates the user’s working memory. There is little or no support to obtain overview information quickly and easily at the beginning of the exploration of a new dataset. This can be a serious limitation when exploring a dataset for the first time.

Overviews become difficult to achieve with large heterogeneous datasets, which is typical in the Semantic Web. As datasets grow bigger and more complex, it becomes more difficult to understand the overall structure and to browse them efficiently [HHG09]. A common approach to obtain an overview and support the exploration of large datasets is to structure them hierarchically [EF10]. Hierarchies allow users to visualize different abstractions of the underlying data at different levels of detail. They can improve the navigation as they allow users to create a mental model of the content structure [vHvW02].

Related to Semantic Web and Linked Data, this overview is usually about which are the main kinds of things in the dataset and how they are structured, i.e. the more instantiated classes and their hierarchical structure. It is also possible to obtain an overview from the point of view of the more common subjects the data is about and how they are structured, for instance as thesaurus.

We propose four components to obtain an overview: navigation menus, site map, site index and treemap. In the following we describe these components.

4.2.1.1. Navigation menus

The Global Navigation pattern proposes to design a persistent set of links that enables users to get to the key areas or functions of a website. These links should be visible to users from every page, so they can move directly from one section to another. Navigation menus, in the case of website, let users navigate through different sections and pages of the site. They tend to be the only consistent navigation element, being present on every page of the site.

4.2. TASKS FOR DATA ANALYSIS

4.2.1.2. HTML Site maps

HTML site maps act as a navigation aid by providing an overview of the site's content at a single glance. They are designed to help users find content on the website by listing all the pages it contains, normally organized hierarchically. In the case of large sites, instead of containing links to all the pages, they can list the main pages (e.g. categories) of the site. When the site contains many levels in the structure and many elements on each level the site map functions as a navigation alternative to navigation menus.

4.2.1.3. Site index

A site index is a navigational and informational tool that lists all the pages or categories alphabetically. While a site map provides a general view of the overall site contents, an A-Z index provides access to particular content. An alphabetical list can better suit users' mental model when they are searching for a specific page.

4.2.1.4. Treemap

Treemaps are one of the most popular methods for displaying hierarchical data [Shn92]. They produce space-efficient overviews of hierarchically structured datasets. They visualize hierarchical relationships of the data elements but also display quantitative information regarding the distribution of elements by their size. Treemaps use a rectangle to show the tree root and its children. Each child has a size proportional to the cumulative size of its descendants. They are very effective when size is the most important feature to be displayed.

4.2.2. Zoom & Filter

After obtaining a basic overview, users want to further explore data. With navigation menus and other overview components we can make the user aware of the main structure of a dataset but, once they choose the class of things they are interested in, they face the barrier of the complexity and size of the data. Users don't always know exactly what they are looking for and, sometimes, they don't even know how it is named or the terms used to describe it. Moreover, they may be unfamiliar with the domain or they want to learn about a topic. Zooming and filtering involves reducing the complexity of data and allows to focus on the resources of interest.

In these cases, exploratory search [Mar06] is a strategy that allows users to refine their search by successive iterations. In exploratory tasks, the user has a vague idea of what he wants to find. His interests change depending on the information context, being similar to the act of berrypicking [Bat89] Exploratory search allows users not only to look up things, but to investigate and learn about the data. These tasks are common in the exploration of semantic data, where users may want to identify the main properties that describe resources, which ones are the most relevant for that particular kind of things, the range of values they have in that particular case, etc.

4.2.2.1. Faceted navigation

Faceted navigation or faceted browsing is an approach to support exploratory search [YSLH03, PSDB99]. An exploratory interface such as faceted browsing allows users to find information without a priori knowledge of its schema. Facets allow users to navigate a collection of elements in multiple ways, rather than a single and pre-determined order [SF09, EHS⁺02, HEE⁺02].

Faceted browser interfaces provide a user-friendly way to navigate through a wide range of data collections. A faceted classification system allows contents to be classified in multiple dimensions. These dimensions are called facets and represent characteristics of the information elements. For example, a collection of books can be classified using an author facet, a subject facet, a publication date facet, etc. In the same way, a collection of movies may be classified by actors, directors, subject, etc. In the Semantic Web, expressed in RDF, resources constitute the collection of browsed elements and facets are the properties that describe them.

Faceted browsing allows to construct complex queries without writing them by hand [Fer08]. In faceted browsing, the constructed query is not returned explicitly. Instead, the user performs an incremental refinement of the results by selecting values of facets that are turned into restrictions. Dynamic queries [AWS92] are one of the keys in faceted browsing. It is fundamental the dynamic update of the display when the user manipulates the filters. Sliders, buttons and other widgets allow users to control the contents and focus on their interests by eliminating uninteresting items. Once the user performs an action, he should immediately see the effect on the display.

When dealing with structured data, facets become a powerful tool for navigation and refining the results. This is a common situation in the exploration of RDF datasets, where users need to identify classes and properties from the schema and learn about the domain. Facet-based filters allow users to formulate semantic queries without requiring any prior knowledge of the data or learning Semantic Web technologies.

Faceted navigation has become very popular in e-commerce applications like eBay (Figure 4.1) or Amazon (Figure 4.2). In the case of RDF, faceted browser interfaces were originally demonstrated in the *Flamenco System* [EII01, Hea00] and have become popular thanks to other projects such as *MUSEUM FINLAND* [HMS⁺05].

4.2.3. Details-on-demand

Limitations of the screen size do not allow to present all the properties and relations of a set of resources by default. The result sets may contain thousands of resources and properties. Presenting all this information of a given resource can lead to information overload [Tho07].

However, once a collection of resources has been reduced to fewer number of items, users should be able to browse the details about the group or individually. To minimise the amount of data fetched and displayed to the user, only additional information is obtained when required by the user.

4.2. TASKS FOR DATA ANALYSIS

The screenshot displays the eBay 'Cars & Trucks Finder' interface. On the left, there are several faceted search filters: 'Make' (set to 'Any'), 'Model' (set to 'Any'), 'Model Year' (with an example range of '1970-2002-04'), 'Distance' (set to 'Any miles of'), 'Far Sale By' (with options for Dealer, Private Seller, and Not Specified), 'Body Type', 'Condition', 'Price' (with a range selector), and 'Vehicle Mileage' (with an option for 'Less than 20,000 miles'). The main search results area shows 38,493 matches. A table lists several results, including a 2007 Ford Mustang GT Supercharged Deluxe 15K Miles Coup., a 2006 Ford Mustang Base Convertible 2-Door, a 2001 Ford Mustang Saleen, and a 1989 Chevrolet Corvette Z51. Each result includes a thumbnail image, a 'Compare' button, and columns for Year, Mileage, Price, and Time Left.

Figure 4.1: Faceted browsing in eBay

The screenshot shows the Amazon 'Movies & TV' page. The top navigation bar includes the Amazon logo, 'Your Amazon.com', 'Today's Deals', 'Gift Cards', 'Sell', and 'Help'. The search bar is set to 'Movies & TV'. The left sidebar contains faceted search filters: 'Format' (with options for Amazon Instant Video, Blu-ray, DVD, and VHS), 'Department' (Movies & TV, TV), 'International Shipping' (Amazon Global, Eligible), and 'New Releases' (Last 7 Days, Last 30 Days, Last 90 Days, Coming Soon). The main content area shows 'Showing 1 - 12 of 756,866 Results'. The first result is 'INSANITY DVD Workout' with a price of \$149.99 (reduced from \$164.99). The second result is 'Duck Dynasty: Season 3 Starring Jase Robertson, Key Robertson, Phil Robertson, et al. (2013)' with a price of \$19.99 (reduced from \$29.99). The third result is '42 (DVD+UltraViolet) Starring Chadwick Boseman, Harrison Ford, Nicole Beharie, et al. (2013)'. Each result includes a thumbnail image, a star rating, and a 'Buy now' button.

Figure 4.2: Faceted browsing in Amazon

4.2.3.1. RDF representation and visualization

An usual approach is to represent the resources as HTML. This is a common feature in many Semantic Web browsers, which is also the case of *Rhizomer*. *Rhizomer* provides an HTML view on the data, displaying all properties and values of a concrete resource. To increase usability, the RDF syntax is completely hidden to the user.

Besides this typical representation of RDF data, we also propose to show specific visualizations depending on the underlying data. For example, those resources with geospatial information can be placed on a map. Visualizations can improve understanding of data, enabling effective knowledge discovery and analytical activity [CMS99].

4.2.4. Relate

Users perform the relate task when they view and follow relationships among resources. This task is related with the filter and details-on-demand tasks.

4.2.4.1. Links to related resources

When exploring Semantic Web data, the way to perform the relate task is by navigating across the graph. This navigation is performed by following links to related resources, normally displayed in the detail view. All those properties whose value is another resource are displayed as HTML links, which users can follow. Moreover, inverse properties can also be used to navigate to related resources. For example, a user obtains details about “Woody Allen”. Then he can see all the properties and values of that resource and follow the links to navigate to related resources. The user can navigate to one of the movies that Woody Allen has directed, e.g. “Vicky Cristina Barcelona”.

4.2.4.2. Set-based browsing

Rather than navigating from a single resource to a single resource, set-based browsing allows users to navigate through data in a more complex way, moving from a set of resources to a related set. For example, a user exploring films recorded in “Spain”, can switch to the set of actors starring in those films. Set-based navigation is related with the functionality of pivoting, which is explained in more detail and implemented in Iteration 2 (Section 7.2).

4.2.5. History

When users explore data, they should be able to return easily to previous states. If the user makes a mistake, he could be able to recover from it. Besides the history provided by web browsers, applications should provide a history of the commands performed to return the interface to a previous state. Comparing the current state with a previous state can result in a better understanding of the data. In addition, history can also support the ability to replay actions.

4.2.5.1. Breadcrumbs

The history task is implemented as Breadcrumbs. Breadcrumbs are navigation components used in user interfaces to keep a track of user’s location or history of actions within a website [Ber88]. Their name comes from the trail of breadcrumbs left by Hansel and Gretel in the fairytale. Breadcrumbs usually appear horizontally below menu bars or headers and they provide a trail for the user to follow back to the starting point.

4.2. TASKS FOR DATA ANALYSIS

Breadcrumbs in websites prevent users from getting lost [Smi96]. They can reinforce the idea that users are in the right place. If users are disoriented [MF95, PK00] they can select one of the previous breadcrumb links to go back to a previous and known point. Then, they can keep with their goal.

Although the breadcrumb metaphor means to mark the specific path the user has taken, there are different kinds of breadcrumbs and some of them do not strictly follow this metaphor. It is possible to distinguish between three types of breadcrumbs [Ins] that can be applied in the Semantic Web context:

Location breadcrumbs

Location breadcrumbs are always static and show where the page is located in the website hierarchy. In the context of the Semantic Web, they indicate the position of a resource in the class hierarchy (or other possible hierarchies such as SKOS concepts hierarchy). They are the simplest and most used because they are easy to implement. Location breadcrumbs are static because a concrete resource has always the same breadcrumb, no matter how users get there.

Path breadcrumbs

Path breadcrumbs are dynamic and show the path the user has taken to arrive to that page. They are the most adequate representation of the breadcrumb metaphor. They indicate how the user got to the current resource and they show the previous resources the user visited before. A concrete resource can have different breadcrumb paths because users can take different routes to get there. They are useful for websites with graph-like structure, which is the case of the Semantic Web.

Attribute breadcrumbs

Attribute breadcrumbs give meta-information that categorizes the current page. They represent the classification of a resource showing which categories it belongs to. A concrete resource can have many attribute breadcrumbs, representing its different possible classifications. Like with location breadcrumbs, users can have several possible paths to reach a resource depending on the properties they use to filter.

4.2.6. Extract

The information that users discover may be important for other tasks and related work projects. Therefore, once users have obtained the resources they are interested in, they should be able to extract important findings. This extraction allows them to save their work and prevents from repeating data manipulations in the future. For tech-users the extraction could be performed as an RDF dump of the data. However, this is not an option for lay-users, who do not understand this format. It would be useful for lay-users to save data in a format that would facilitate printing it, sending by email or reusing it in other applications.

4.2.6.1. Bookmarks

To perform this task we do not propose any specific Information Architecture component. Users have the possibility to simply bookmark the page with their browser for later revisits or sharing it with other users. In the context of the Web, a bookmark is a URI that can be stored for later retrieval. Nowadays all modern web browsers include bookmark features.

5.1. MPlu+a

The methodology followed in this project is based on the MPlu+a development process [Gra03]. MPlu+a is a development framework for interactive systems that integrates the discipline of Software Engineering with the basis of Human-Computer Interaction, Usability and Accessibility.

5.1.1. Overview

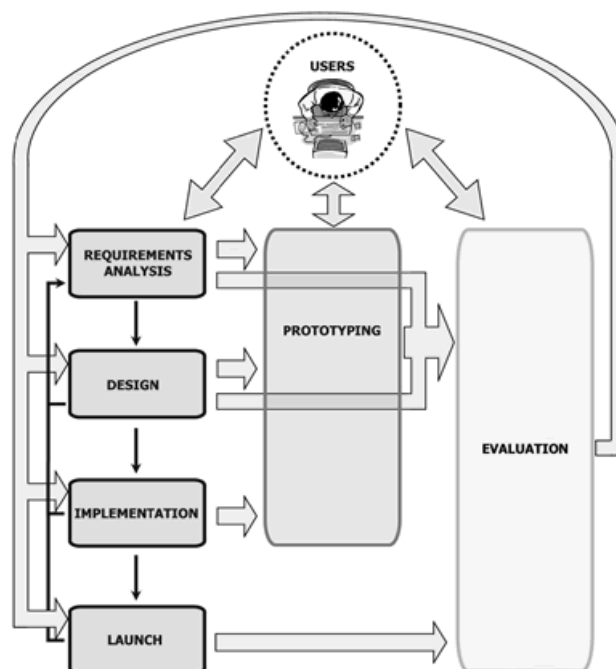


Figure 5.1: MPlu+a organization, from <http://www.grihocitools.udl.cat/mpiua>

Figure 5.1 shows the process model schema with its phases and the relation between them. These are the main features of this process model:

- **Conceptual organization:** the schema is organised in modules or stages that show the current phase.
- **Three main pillars:** the schema shows these three main pillars in different colors:
 - Software engineering: the classic software development life cycle based on the waterfall model (left column).
 - Prototyping: grouping all the techniques to build software samples to facilitate the subsequent evaluation (center column).
 - Evaluation: covering all the usability and accessibility validation methods.
- **The user:** a User Centred Process Model has the user as the most important part. This schema reflects this meaning at the first glance, placing the User in central and above the other phases.
- **An iterative model:** the schema has a series of arrows to show the relation between phases and users' active participation in some of them: requirements analysis, prototyping and evaluation.

5.1.2. User-Centered Design

User-Centered Design (UCD) is a type of design process for interactive systems focused on the users who will use the system. In UCD the user participates in all the stages of the design process. User requirements are considered from the beginning and included into the whole development cycle. These requirements are defined and refined through different methods such as ethnographic studies, focus groups, usability testing, etc.

It is important to highlight that User-Centered means focusing on all users. This implies considering all their differential characteristics and also thinking about those with a disability [Ste95].

The ISO 13407 standard [Int99] establishes a framework that provides guidance to achieve the development of usable interactive systems incorporating the UCD during the development life cycle. This standard describes the User-Centered Design as a multidisciplinary activity that also includes human factors and ergonomic techniques.

5.1.2.1. Usability

The concept of usability is defined as the ease of use and learnability of a human-made object. Usability is a property that can be applied not only to software systems, but also to elements of our everyday life [Nor90].

In software systems, the concept of usability was introduced by J. Nielsen [Nie93] as a quality attribute that assesses how easy user interfaces are to use. The ISO 9241-11 standard [ISO98] defines usability as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use."

Usability is defined by 5 quality components:

- **Learnability:** how easy is it for users to accomplish basic tasks the first time they use a system.
- **Efficiency:** how quickly can users perform tasks once they have learned the system's design.
- **Memorability:** how easily can users reestablish proficiency when they return to a system after a period of not using it.
- **Errors:** how many errors do users make, how severe they are, and how easily they can recover from them.
- **Satisfaction:** how pleasant using the system is.

5.1.2.2. Accessibility

Accessibility is the degree to which a product or service is available to as many people as possible. Nowadays, providing accessibility has become an important factor in interactive systems. The ISO standard [IfS03] provides a guide with the ergonomic specifications to design computer interfaces.

Web accessibility [wai] means that everyone can use the web, no matter their possible disabilities. The concept of web accessibility is often used to focus on people with disabilities or special needs. However, web accessibility can benefit everyone; old people or people with temporary disabilities (vision problems, injuries, etc.) can also be benefited from it. Web accessibility can also improve the user interaction. The improvements in usability, navigation and content structure benefit everyone.

5.1.3. Software engineering

5.1.3.1. Requirements analysis

Requirements analysis in software engineering covers all the tasks to determine the user needs and conditions regarding to a concrete software or product. This phase is very important in every development and necessary to obtain good final results. In this way, the number of errors is reduced because most factors had already been considered in this phase.

Requirements analysis can be divided into the following activities [SK98]:

- **Eliciting requirements:** collecting the requirements from users and other stakeholders using different techniques.
- **Analyzing requirements:** determine whether or not the requirements are correct, consistent and resolve possible conflicts between them.
- **Documenting requirements:** a good documentation will facilitate its further implementation. Requirements can be documented in various forms such as use cases, specifications, etc.

- Validating requirements: make sure that the system supports all the requirements correctly.

Software requirements give a complete description of the system to be developed. They describe what a system must do and how it must be done. The traditional software engineering distinguishes between two types of requirements:

- Functional requirements: describe the system functionalities.
- Non functional requirements: describe other restrictions of the system (e.g. response time) or about how its development must be (e.g. use a specific programming language).

5.1.3.2. Design

Once obtained the requirements, the second phase in the MPlu+a process is the design. Software design is the process of problem planning for a software solution. It includes the planning of algorithms as well as the architecture of the system.

One of the most important parts of interactive systems is the dialog between the user and the system. The user interface is the part of the system that allows the user to interact with it. The user interface determines the user's perception and conception of the application [Thi90].

The interaction design is divided into two activities:

- Activity design: the activity design includes the analysis of functionalities and tasks that users can carry out.
- Information design: related with the interface layout, its components and their style.

An important aspect in this phase is the affordance. The affordance or intuitive comprehension [Nor90] is the ability of an element of the user interface to give the impression that it can be correctly used.

5.1.3.3. Implementation

Usability Engineering and the MPlu+a process model do not specifically describe this phase since it belongs to the classic software development process. The final product is created in this phase. Developers must select the programming languages, database systems and the best technologies that fit the system.

5.1.3.4. Launch

Software launch or release is one of the most critical phases in the development process. The product's success will depend on two important factors:

- The degree to which the user is comfortable with the system: the system errors, its simplicity, its functionalities, etc.

- The degree to which the project responsible persons obtain the expected results.

The MPIu+a process model helps these factors to be satisfied since the design has been done focusing on users and for users. They have been involved in all the process.

5.1.4. Prototyping

Prototypes are documents, designs or systems that are incomplete versions of a software program being developed. They simulate or implement parts of the final system [RC02], that can be different from the final product.

Prototyping has several benefits: the final users can get involved in the development process and the software designers can get feedback from them. With prototypes it is possible to evaluate the product from the beginning of the development. Prototyping can also be used to obtain requirements that were not considered previously [Bro95].

Nielsen describes two dimensions of prototypes [Nie93]:

- **Horizontal prototypes:** they provide a broad view of the entire system and its interface but they do not implement much functionalities. They focus on the user interaction more than in the system's functionality. An horizontal prototype is a simulation of the interface in which no real tasks can be done [HL90].
- **Vertical prototypes:** they provide a more complete elaboration of a part of the system. A vertical prototype can prove a limited part of the system but under real circumstances.

Prototyping techniques can be classified according to the fidelity with which they resemble the actual product in terms of appearance, interaction and timing [Ret94]:

- **Low fidelity:** they implement general features of the system without going into details. They are economical, easy to build and they don't require the use of any specific tools.
- **High fidelity:** they represent more precise features of the system. They can detail specific tasks or features. They are more expensive, they require more time and the use of specific tools.

Some of the prototyping techniques are: sketching, storyboards, paper prototypes, navigational storyboards, software prototypes, etc.

5.1.5. Evaluation

Evaluate consists in proving something to know if it works correctly, if it covers the expectations or just to see how it works. Usability evaluation is a major aspect in any UCD methodology. Usability evaluation covers all the methodologies and techniques to analyze the usability and accessibility of a product.

In the MPIu+a process model, the evaluation phase is the key to obtain usable and accessible systems. Evaluation must not be considered as a single step in the development process, but should be practiced throughout the whole lifecycle.

According to Dix [DFAB97], evaluation has three main objectives:

- Check the system functionalities.
- Check the user interface and its effect on users.
- Identify any other specific problem related with the system.

Usability evaluation methods can be classified in three categories:

- **Inspection methods:** inspection is the generic name for a set of methods where an expert evaluator inspects a user interface. Inspection methods have different objectives but all them are based on experts' opinions and reports [Nie95]. Two examples of inspection methods are heuristic evaluation [NM90] and cognitive walkthrough [WoCloCS92].
- **Inquiry methods:** they involve collecting qualitative data from users. These methods require observing and interviewing users to know their opinions, needs or complaints about the system. Inquiry methods include interviews, focus groups, questionnaires, etc.
- **Test methods:** in test evaluation methods end users perform concrete tasks using the system or a prototype. Sessions are usually recorded on video. Evaluators collect the most quantitative data such as task completion time, task completion rate, etc. The think aloud protocol [Lew82] is often used to know participants' thoughts on the application while executing tasks.

5.2. SWET-QUM

All software developers should aspire to achieve a high level of quality in software systems. In order to make Semantic Web tools more appealing to lay-users, a key factor is their Quality in Use, i.e. the degree to which a product or system can be used to achieve user's goals in specific contexts of use. To assess and motivate the improvement of the quality in use, it is necessary to have a quality model that guides its evaluation and facilitates comparability.

So far, in the Semantic Web context, quality in use has received less attention than other quality aspects. The main focus of quality models has been normally placed on internal quality, what makes it possible to build good Semantic Web applications. However, there is far less work related with building standards-based quality models to evaluate Semantic Web technologies. As more applications are developed and more users start using them, aspects related to external quality are getting more and more relevant as interest spreads from building Semantic Web applications to also getting users to use them.

We propose a Semantic Web Exploration Tools Quality in Use Model (SWET-QUM) to evaluate the quality of applications based on Semantic Web technologies. Since it is a very broad task, we have focused on a particular aspect of quality: the external part related with the quality in use. Moreover, we have also restricted the scope of this model to a subset of Semantic Web applications. The applications under consideration are those that allow to explore Semantic Web data to end-users with potentially no knowledge about these technologies. These applications are referred as Semantic Web Exploration Tools (SWETs). Moreover, as we are also developing one of these tools, *Rhizomer*, we have tested and improved the proposed quality model during its development.

5.2.1. The concept of quality

The Quality of a product or service can be defined as its “rightness to meet user needs and the degree to which a set of inherent characteristics fulfils requirements” [Int00]. Another definition of Quality can be: ‘conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software’ [Pre05].

These definitions of quality reveal two considerations. The first one is the need to characterize the concept of quality based on the identification of the inherent properties of the product (quality of a product), i.e. the definition and development of a Quality Model. The other one is the need to establish or propose a series of functional and/or non-functional requirements, and how these are achieved by users through the interaction process or process of use (quality in use).

The decomposition of quality in other features makes the process of quality evaluation easier. Basili [BW84] describes a quality model based on three key components:

- Factors or characteristics (to specify): indicate which properties and targets are used as indicators of the quality of a product. These properties identify desirable aspects or characteristics of software product quality. They describe the external view of the software, as viewed by the users.
- Criteria or properties (to build): indicate measurable attributes linked to the factors of a software product. These properties can be evaluated through direct and indirect metrics. They describe the internal view of the software, as seen by the developer.
- Metrics (to control): determine the evaluation of a software product and allow to estimate its features. The metrics are defined and used to provide a scale and method for measurement.

According to the international standards, the quality of an interactive system has two principal components or dimensions. One is the product component with internal and external points of view. This component has special relevance in Software Engineering disciplines. The other quality component is focused on how the users use the functionality/performing tasks in a specific context of use (effect of software product). This component can be characterized by properties such as usability in use, flexibility in use or freedom from risk. All these properties are related to the Human-Computer Interaction discipline [Bev01] and new standards treat it as a quality measure itself. It is known as quality in use, as shown in Figure 5.2.

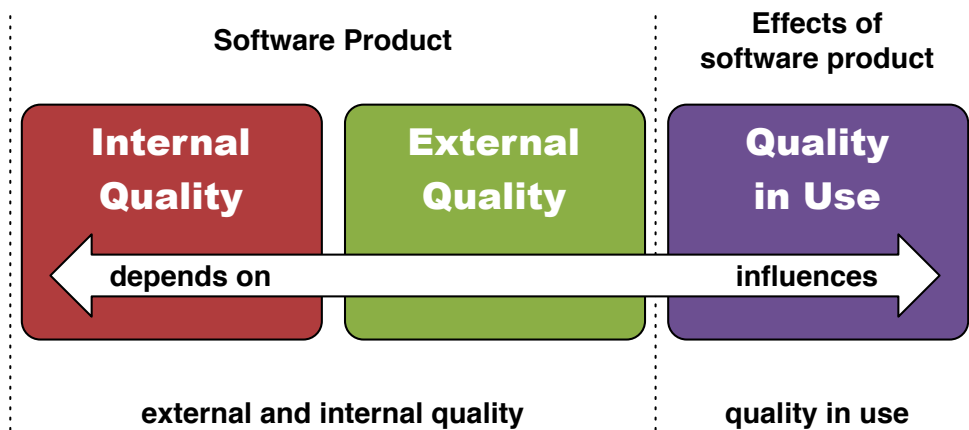


Figure 5.2: Quality of a software product

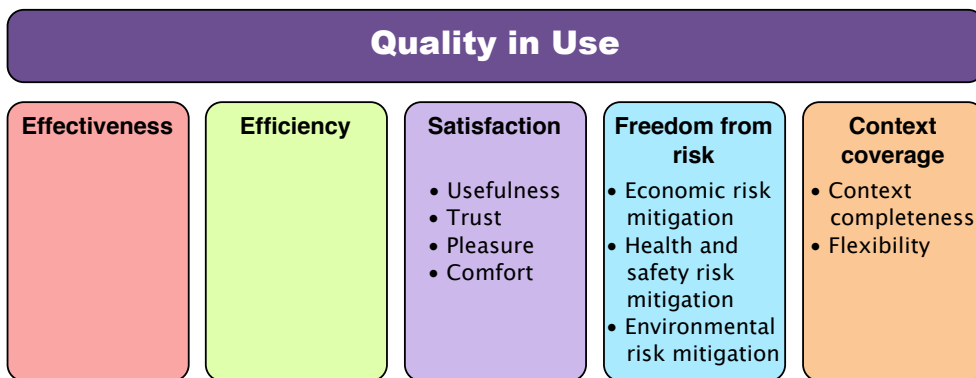


Figure 5.3: Quality in use factors

5.2.2. Quality in Use for SWETs

Many standard models exist, but we focus on and build from the latest ISO/IEC standard model, ISO/IEC 25010:2011 [Int11]. It provides guidance for the use of the new series of international standards named Software product Quality Requirements and Evaluation (SQuARE). This standard replaces the old ISO/IEC 9126 [iso01] and comprises the second generation of standards for software quality.

This international standard defines a Quality in Use model composed of five factors/characteristics related to the outcome of interaction when a product is used in a particular context of use. This system model is applicable to the complete human-computer system, including both computer systems in use and software products in use. The five factors are: effectiveness, efficiency, context coverage, freedom from risk and satisfaction, as shown in Figure 5.3.

5.2. SWET-QUM

We propose a Semantic Web Exploration Tools Quality in Use Model (SWET-QUM) that specialises the generic Quality in Use characteristics/factors and properties proposed in ISO/IEC-25010:2011 [Int11], for the evaluation of quality of interaction for Semantic Web exploration tools. We consider all the characteristics for Quality in Use in ISO/IEC-25010:2011 except for Freedom from Risk, which includes aspects like economical, health and environmental risks. This factor is more appropriate when considering ergonomic and other related factors that, for the moment, lay out of the proposal scope.

SWET-QUM is completed with metrics focused on process of use. Hence, the evaluation essentially requires testing with users, observation of users while they are interacting and the completion of questionnaires to measure satisfaction when they finish the tasks. Our proposed metrics have an interpretive approach and are focused on the resolution of tasks. Consequently, they are based on users interaction towards achieving test tasks goals.

The following subsections present the Quality Factors considered in SWET-QUM. It is described how they have been interpreted in the context of the proposed model and the metrics used to measure the properties corresponding to each factor. Some of the metrics are selected from the standard [cif04] and then an estimation formula is proposed as detailed in the next subsections. Moreover, additional metrics are also proposed. They focus on the SWET scenario and also cover a factor less considered in the standard but quite relevant in the case of SWET, i.e. context coverage.

5.2.2.1. Effectiveness

Effectiveness is defined as *the degree to which specific users can achieve the semantic data exploration tasks with precision and completeness*. The metrics under consideration for Effectiveness in the SWET-QUM include the common ones for measuring the effectiveness of the tasks posed to the users, i.e. task success for the effectiveness of an individual task and tasks completion when considered the set of tasks included in one evaluation session. Moreover, we have also added one measure for the effectiveness of the user interface components responsible for supporting user interaction during data exploration tasks. The proposed metrics are:

1. **Task success:** to what degree an individual task posed to the user during the user test is completed?
 - **Measure:** $X = F(X)$ (percentage of the task completed).
 - **Value:** $0\% \leq X \leq 100\%$ (0% means that the task was not completed at all, usually when the user gives up the proposed task. 50% means that the user completed the task partially and 100% that the task was fulfilled in completely).
 - **Input:** operation (test) report. User monitoring record.
2. **Tasks completion:** what proportion of all the tasks posed to the user during the test is completed? This metric provides an overview of the overall effectiveness attained by the user during a test session.
 - **Measure:** $X = 100 * A/B$ (A is the number of tasks completed and B the total number of tasks attempted).

- **Value:** $0\% \leq X \leq 100\%$ (0% means none of the tasks was completed and 100% that all the tasks were completed).
 - **Input:** operation (test) report. User monitoring record.
3. **Data Exploration UI Effectiveness:** what proportion of the user interface components, relevant for the task, do the users view? These components are those relevant for the data exploration tasks and include high level components like menus, facets, breadcrumbs, etc. but also other components when a more detailed view is necessary: links, buttons, forms, etc.
- **Measure:** $X = 100 * A/B$ (where A is the number of relevant components viewed by the users and B the total number of relevant components).
 - **Value:** $0\% \leq X \leq 100\%$. (0% means that none of the relevant UI components received user attention while 100% means that all the relevant components did receive, at some time point during the user test, the attention of the user).
 - **Input:** operation (test) report. User monitoring record. Eye tracking of the screen areas corresponding to each of the monitored components. A component receives the attention of the user if the Eye Tracker reports that the specified screen area occupied by the component received user gaze during more than 60ms [KGJ⁺10].

5.2.2.2. Efficiency

Efficiency is defined as *the degree to which specific users can achieve the proposed tasks by investing an appropriate amount of resources in relation to the effectiveness achieved in a semantic data exploration context of use*. This factor is determined by the ease of learning and the ease of interaction with data. Classical efficiency measures have been considered, i.e. the time to complete a task and the total time for all tasks. These pure efficiency measures have been complemented with one that relates efficiency to task success: task efficiency. Moreover, due to the complexity of the tools under consideration and that the target are lay-users, it is also important to take into account the requests for help that users ask to facilitators when they get stuck with the task. Finally, like in the case of effectiveness, we have also considered to measure the efficiency of the user interface components responsible for supporting user interaction during data exploration tasks. The metrics for these five measures are:

1. **Task time:** how long does it take to complete an individual task? In other words, how much time does the user invest to complete the task?
 - **Measure:** $X = Ta$ (where Ta is the task time).
 - **Value:** $0 \leq X$ (the smaller the better).
 - **Input:** operation (test) report. User monitoring record.
2. **Total time:** how long does it take the user to complete all the tasks posed during the test session?
 - **Measure:** $X = TT$ (where TT is the total time).
 - **Value:** $0 \leq X$ (the smaller the better).
 - **Input:** operation (test) report. User monitoring record.

3. **Task efficiency:** how efficient are the users in comparison with task success?
 - **Measure:** $X = M / T$ (where M is task success and T is task time).
 - **Value:** $0 \leq X$ (0 corresponds to a non-successful task and 100 to a successfully completed task in 1 minute).
 - **Input:** operation (test) report. User monitoring record.
4. **Facilitator help requests:** how many help requests has the user asked to the facilitator? The time spent in doubt formulation and clarification is considered part of the task time.
 - **Measure:** $X = Rf$ (where Rf is the number of help requests to the facilitator).
 - **Value:** $0 \leq X$ (0 means the user did not ask any help request).
 - **Input:** operation (test) report. User monitoring record.
5. **UI Component Efficiency:** what percentage of the attention of the user is captured by the components relevant for data exploration tasks? The percentage is relative to the total time spent with the components. The objective in this case is to check if the most part of the UI for data exploration is used so there are no parts of it that are occupying UI space but are not considered by the user. The same kind of components that in Data Exploration UI Effectiveness are considered in this case.
 - **Measure:** $X_i = T_{Ai} / T$ (where A_i is the time spent looking at relevant UI component i and T the total time spent looking at all the relevant components).
 - **Value:** $0 \leq X_i \leq 1$ (when closer to 1 the more attention has been paid to component i).
 - **Input:** eye tracking record of the screen areas corresponding to each of the monitored components.

5.2.2.3. Context Coverage

Context coverage is defined as *the degree to which the Semantic Web exploration tools can be used with efficiency, effectiveness and satisfaction in a specific context of use (context completeness); and how the system can be used in different contexts and adapt to different user mental models (flexibility in use) offering the best user experience*. It typically includes how adaptable the tool is to different usage scenarios (device, user preferences, data, etc.). However, in the case of Semantic Web exploration tools, it is even more relevant to evaluate how well the tool adapts to different datasets.

From the Quality in Use perspective, this means that the tool is capable of offering as many as possible ways to complete the data exploration tasks, independently from the datasets being considered and exploiting in each case the whole richness of the dataset structure. A flexible tool should provide the interaction means to complete the task following the reasonable user mental models for the domain at hand. It should support all the reasonable relationships among entities, even when they are not explicit in the underlying data.

Therefore, we have considered here a measure that relates the number of reasonable ways to complete a particular task, considering the conceptual domain where the task takes place, and the number of ways actually offered by the tool, i.e. task flexibility. This metric is particularly useful to evaluate the Overview, Zoom and Filter user tasks, which are usually combined to reach the set of results the user is interested in and potentially through different paths. We complement this metric with another related to the layout flexibility of the user interface components. This metric is particularly useful for evaluating the Overview user task and check if the user interface allows users to get a quick idea about the information available.

1. **Task Flexibility:** what proportion of the alternative ways to complete the task can be achieved using the tool?
 - **Measure:** $X = A / B$ (where A is the number of alternative ways of completing the task offered by the analysed tool and B the total number of conceivable ways of completing it taking into account the conceptual domain of the task).
 - **Value:** $0 \leq X \leq 1$ (the closer to 1 the better).
 - **Input:** expert analysis of the task, the task domain and the tool user interface.
2. **Layout flexibility:** for a given context of use, what is the average number of interaction steps required to reach the user interface components relevant for the task? For navigation menus this is equivalent to how deep relevant menu options are in the Information Architecture. Consequently, if the component is directly visible for the user, it is considered to be at depth zero. These components might be menu options, forms, facets, etc. The metric checks that for a particular dataset, user interface components are arranged so the most relevant choices are more evident.
 - **Measure:** $X = \Sigma(D_i)/n$ (where D_i is the number of interaction steps required to reach task-relevant UI component i and n is the minimum number of interaction steps required to complete the task).
 - **Value:** $0 \leq X$ (the closer to 0 the better).
 - **Input:** expert analysis of the tool's user interface.

5.2.2.4. Satisfaction

User Satisfaction is defined as *the degree to which users are satisfied when using the data exploration tool*. This factor considers various attributes such as fun, pleasure, comfort, attractiveness, motivation, emotion or sociable sociability (hedonic factors). In this case, we consider a classical way to measure satisfaction, based on questionnaires:

1. **Satisfaction questionnaire:** how satisfied is the user with specific software features?
 - **Measure:** $X = \Sigma(A_i)/n$ (where A_i is the value of the response of user i to the question and n is the number of users that responded).
 - **Value:** compare with previous values or with population average.
 - **Input:** operation (test) report. User monitoring record. User test plus questionnaires.

5.3. Development and evaluation process

SWET-QUM is integrated together with MPlu+a as part of the *Rhizomer* development process. The iterative development process is based on The Rapid Iterative Testing and Evaluation method (RITE) [MWT⁺02]. Documented by researchers at Microsoft, the RITE method proposes a variation of traditional usability testing performing short iterations. Evaluations differ from traditional ones mainly in the sense that much smaller groups of users are recruited for the tests. However, tests are performed much more frequently and it advocates that changes to the user interface are made as soon as a problem is identified and a solution is clear.

Consequently, results for individual test iterations are less significant from a statistical and quantitative point of view. The main results from a testing session are basically qualitative and are used to guide the next development iteration. However, as many evaluation iterations are accumulated along the development process, it is possible to perform a quantitative analysis of the results. Moreover, the overall costs of the evaluation are significantly reduced.

Each iteration, or reduced set of iterations, includes an evaluation of the Quality in Use with a small group of users. We have tested and improved the proposed quality model during the development of *Rhizomer*. Consequently, not all the proposed metrics have been collected in all iterations. In each iteration we consider those metrics that are more appropriate for the objectives and tasks to perform. It is also important to highlight that SWET-QUM has also been used to evaluate other SWET tools: *SParallax* and *Virtuoso Faceted Browser*, as detailed in iteration 2 (section 7.2).

In each iteration, the evaluation process is performed through a mix of evaluations and questionnaires. The evaluations with users are based on tasks to be completed using the tool being evaluated. The interaction is analysed and the selected metrics among the proposed set are used to measure the quality factors of each task. The evaluations with users are complemented with questionnaires that measure the Satisfaction factor and collect information about users' perception or the process of use, i.e. the hedonic and subjective quality. These techniques are organised following the standard Common Industry Format for Usability report (CIF) [cif04] into the following four stages:

5.3.1. Pre-test

The evaluations are conducted at the UsabiliLAB⁸, the usability laboratory of the Universitat de Lleida. The evaluation equipment is based on two computers. One of them is for the user and it is equipped with Morae⁹ Recorder, which registers user interaction, screen video, clicks, mouse position, user voice and user video through a web-cam. The second computer is equipped with Morae Observer and Morae Manager, which are used by the evaluation team to observe, annotate and analyse the interaction session. If it is necessary, there is also a third computer equipped with an eye tracking device that can be used to obtain additional metrics.

⁸UsabiliLAB, <http://griho.udl.cat/about/infrastructure.html>

⁹Morae©<http://www.techsmith.com/morae.html>

In this stage, the test facilities are set and the context of use is defined, including the factors and properties to be measured, the kind of tasks and the users, who are recruited. Then, they sign a confidentiality document, giving permission to be recorded. The user profiles are determined using questionnaires about age, skills, etc. Relevant characteristics about users' profile are: gender (male or female), age, education (number of years of completed formal education) or product experience (type and duration of any prior experience with the product or similar products). Appendix A includes the confidentiality document used in the tests, as well as other documents such as post-task questionnaire and post-test questionnaire.

5.3.2. Test

Participants are introduced to the project and the purpose of the evaluation. After this short introduction, the test facilitator presents the tasks to the users. The tasks are not necessarily performed in the same order to minimize the learning effect. The interaction process is analysed using Morae Observer and Morae Manager to compute the selected quality in use metrics.

During all the evaluation the think-aloud protocol [Lew82] is used. This method used in usability tests proposes that participants express their thoughts on the application while performing test tasks.

5.3.3. Post-test

User satisfaction is measured after performing the test using questionnaires that capture the subjective vision of the interactive process of use. In addition to the post-test satisfaction questionnaires, it is also possible to use post-task satisfaction questionnaires, which are presented to the user after the completion of each individual task.

5.3.4. Reports

After completing the evaluation process, the data resulting from the evaluations with users and questionnaires is analysed. The metrics for the quality factor are computed and they are interpreted in the context of the evaluation session objectives.

Part IV

Contribution

Automatic Information Architecture Generation Methods

In the previous chapters we have outlined the principles of the Semantic Web with its underlying technologies and we have presented the different tasks that can be performed when interacting with Semantic Web data. We have also proposed a set of Information Architecture components to perform these tasks. However, traditional approaches need to be adapted to the size and characteristics of semantic data. The size and heterogeneity of semantic datasets make it impractical to apply traditional and manual Information Architecture generation techniques [RM02]. Therefore, this generation process should be automatized.

In this chapter we present a series of algorithms and methods to automatically generate and drive the information architecture components previously described. They belong to the three basic tasks identified by Shneiderman [Shn96]: overview, filter and details-on-demand.

For the overview task, we propose an algorithm that obtains the class or topic hierarchies structuring a dataset and then generates a navigation menu for it. It allows the users to get an idea about the overall structure of the dataset and serves as a starting point for navigation. The class or topic hierarchies are also used by other overview components: site map, site index, and treemap.

For the filter task, we review methods for ranking facets and we propose a new one. Our method incorporates existing heuristics but also considers the descriptive value of properties. This method is used in our faceted browser to select those facets that best describe a dataset.

Finally, for the details-on-demand task, we propose the Linked Data Visualization Model (LDVM), which allows to dynamically connect data with different visualizations. In order to achieve such flexibility and a high degree of automation, the LDVM is based on a visualization workflow incorporating analytical extraction and visual abstraction steps.

Note: these prefixes have been used in the following sections in order to shorten SPARQL examples. ¹⁰ ¹¹ ¹²

¹⁰PREFIX rdf: <<http://www.w3.org/1999/02/22-rdf-syntax-ns#>>

¹¹PREFIX rdfs: <<http://www.w3.org/2000/01/rdf-schema#>>

¹²PREFIX owl: <<http://www.w3.org/2002/07/owl#>>

6.1. Overview generation and storage

The overview components are generated from the hierarchy structure of the dataset, i.e. its schema. For each class, we store information about the number of instances of the class, its URI, its label and a list of its subclasses. All this information is retrieved using a set of SPARQL queries. Example 6 shows the SPARQL query to obtain the root classes, i.e. those non-blank without a superclass different from `owl:Thing` or `rdfs:Resource`. Example 7 presents the SPARQL query to get direct subclasses for a given class, i.e. there is not an intermediary class between them in the hierarchy. Finally, Example 8 shows the SPARQL query to obtain the number of instances of each instantiated class. For datasets without a schema it is not possible to generate the class hierarchy, only to obtain the list of classes and number of instances for each class. A similar approach can be used to generate a navigation menu with SKOS concept hierarchies [MPA07].

```

1 SELECT DISTINCT ?root
2 WHERE {
3   ?root rdf:type ?class.
4   FILTER (?class=owl:Class || ?class=rdfs:Class)
5   OPTIONAL {
6     ?root rdfs:subClassOf ?super .
7     FILTER (?root!=?super && ?super!=owl:Thing && ?super!=rdfs:Resource && !
8       isBlank(?super))
9   }
10  FILTER (!bound(?super) && isURI(?root) &&
11  !isBlank(?root) && ?root!=owl:Thing )

```

Example 6: SPARQL query to obtain the root classes

```

1 SELECT DISTINCT ?sub
2 WHERE {
3   ?sub rdfs:subClassOf ?classURI .
4   OPTIONAL {
5     ?sub rdfs:subClassOf ?sub2 .
6     ?sub2 rdfs:subClassOf ?classURI .
7     FILTER (?sub!=?sub2 && ?sub2!=<%1$s>
8       && !isBlank(?sub2))
9   }
10  FILTER (!bound(?sub2))
11 }

```

Example 7: SPARQL query to obtain direct subclasses for a given class

```

1 SELECT ?class COUNT(?x)
2 WHERE {
3   ?x a ?class
4 } GROUP BY ?class

```

Example 8: SPARQL query to count the number of instances of each instantiated class

Once the class hierarchy is generated, we store it in a local file so it can be reused by the overview components. If the dataset is modified, i.e. resources are added or removed, this file is updated. We use the VoID vocabulary [AH09] to describe the dataset, its class hierarchy and the number of instances of each class. The `void:classPartition` property is used to provide descriptions of parts of the dataset, i.e. subsets. The `void:distinctSubjects` property expresses the number of distinct subjects (number of instances) that occur in the dataset or subset. By using these properties, each class-based partition describes a subset of instances that belong to a particular class. Each class-based partition can also have nested partitions, which allow to represent the class hierarchy. The VoID file also provides metadata about the dataset that can be reused by other applications and users. Example 9 shows a fragment of the VoID description generated following this approach for the DBpedia dataset.

6.1. OVERVIEW GENERATION AND STORAGE

```
1 <rdf:RDF
2   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3   xmlns:void="http://rdfs.org/ns/void#"
4   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
5   <rdf:Description rdf:about="http://rhizomik.net/dataset">
6     <void:classPartition>
7       <rdf:Description rdf:about="http://dbpedia.org/ontology/Species">
8         <rdfs:label>Species</rdfs:label>
9         <void:distinctSubjects rdf:datatype="http://www.w3.org/2001/XMLSchema#
10          integer"
11          >202339</void:distinctSubjects>
12       <void:classPartition>
13         <rdf:Description rdf:about="http://dbpedia.org/ontology/Archaea">
14           <rdfs:label>Archaea</rdfs:label>
15           <void:distinctSubjects rdf:datatype="http://www.w3.org/2001/XMLSchema#
16            integer"
17            >171</void:distinctSubjects>
18         </rdf:Description>
19       </void:classPartition>
20     <void:classPartition>
21       <rdf:Description rdf:about="http://dbpedia.org/ontology/Bacteria">
22         <rdfs:label>Bacteria</rdfs:label>
23         <void:distinctSubjects rdf:datatype="http://www.w3.org/2001/XMLSchema#
24          integer"
25          >328</void:distinctSubjects>
26       </rdf:Description>
27     </void:classPartition>
28   <void:classPartition>
29     <rdf:Description rdf:about="http://dbpedia.org/ontology/Eukaryote">
30       <rdfs:label>Eukaryote</rdfs:label>
31       <void:distinctSubjects rdf:datatype="http://www.w3.org/2001/XMLSchema#
32        integer"
33        >199085</void:distinctSubjects>
34     </rdf:Description>
35   </void:classPartition>
36 </rdf:Description>
37 </rdf:RDF>
```

Example 9: Part of the VoID description for the DBpedia dataset

This information is reused by the overview components implemented in Iteration 1 (Section 7.1) and Iteration 3 (Section 7.3).

6.1.1. Algorithm to generate navigation menus

Navigation menus can only provide an overview of the most important pages because they have a limited space. In our case, considering most semantic datasets, there might not be enough space to display all the relevant classes in the hierarchy. Alternatively, datasets may have only a few classes in the first level of the hierarchy and most of the space would be wasted. Therefore, navigation menus should be adapted to the number of classes and size of the dataset in order to make a better use of the available space.

The objective is to automate this process and generate a global navigation menu that takes into account all the classes in a dataset but also how they are instantiated. Consequently, if there are few instances of some classes or they are not instantiated at all, they should be less relevant in the menu bar. On the contrary, classes that have a lot of instances should be shown prominently in the menu bar. In this way, the menu facilitates the access to the more significant classes but also makes it possible for new users to realise what are the main kinds of things in a dataset.

This approach makes it possible to show the user the navigation bar that best fits the data in the dataset at that particular moment. For instance, if the dataset changes from containing mainly data about projects to mainly about publications, the menu would change accordingly to show more prominently the part of the dataset structure about publications.

On the other hand, one possible drawback of this approach, as it has been pointed by some usability expert evaluations [GGP⁺10], is that users find it very disturbing that the navigation menus change from visit to visit due to changes in the underlying data. This is an inconvenient effect of navigation menus dynamism, as users see them as a static part of the site and, as they get used to them, they rely on them as a handful guide to the site.

Anyway, our experiments show that these changes are only perceivable for small datasets. Under those circumstances, the navigation menu undergoes changes quite often when adding new resources. However, as more resources are introduced, changes in the navigation menu tend to be minimal. As soon as the amount of data is statistically significant to keep the natural tendency in the dataset evolution, the changes in the menu bar are practically inexistent or not significant from the point of view of the user. They only affect to particular options in the submenus that are added or removed in the context of more general options in the menu, that keep users in the track to the information they need.

The navigation menus are created from the hierarchy structure generated. This component can generate both global and local menus, i.e. a menu for the whole dataset or for a subset of it. The site administrator can also configure some parameters:

- The number of levels in the hierarchical menu.
- The number of items in each level of the menu.
- The order of items: alphabetically or by number of instances.
- A list of classes or namespaces to omit.

According to these parameters, this component generates the menu applying a recursive algorithm, shown in Example 10, that mainly performs two operations:

- Split those classes with a large amount of instances in subclasses.
- Group those classes with few instances in a superclass.

```
1 generateMenu(Menu menu, int numItems) {
2   menu.removeEmpty();
3   while(menu.size()>numItems) {
4     Node other = menu.createOther();
5     Node min = menu.getMinNode();
6     other.mergeWith(min);
7   }
8   while(menu.size()<numItems){
9     Node max = menu.getMaxNode();
10    menu.splitNode(max);
11  }
12 }
```

Example 10: Overview of the navigation menu generation algorithm

6.1. OVERVIEW GENERATION AND STORAGE

The algorithm starts with a menu tree structure that initially contains the whole hierarchy of classes and the number of instances for each class. The first step of the algorithm is to remove all the empty classes that have zero instances. Then, depending on the number of intended items in the final menu, i.e. parameter “numItems”, the algorithm performs mainly two operations:

- If the number of menu items is higher than the input parameter, those classes with fewer instances are grouped in a new class called “Other”.
- If the number of menu items is smaller than the input parameter, the class with more instances is divided into its subclasses.

These operations are recursively performed until the menu is completed. Figure 6.1 illustrates the process of generating the navigation menu for a subset of DBPedia 3.5, with 7 elements in the first level. In the original hierarchy there are only 3 classes in the first level. Therefore, there are 4 free spots in the menu. To cover these free spots, the algorithm identifies which classes are appropriate to divide, taking into account their number of instances and their number of subclasses.

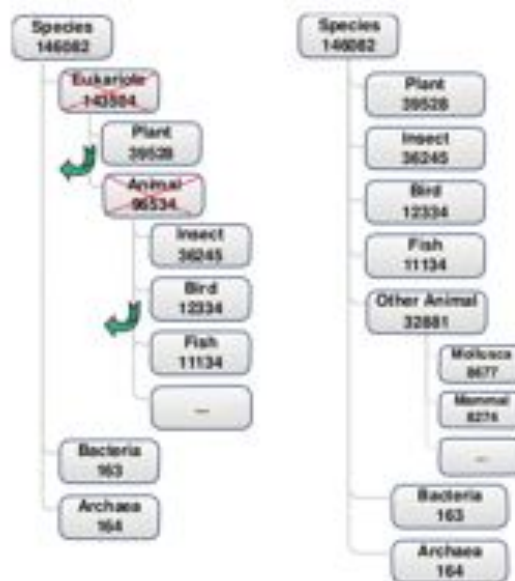


Figure 6.1: Generating a navigation submenu for DBPedia Species with 7 slots (left original, right result)

At first, the Eukariote class is removed and its subclasses, Plant and Animal, move up to a higher level in the hierarchy. After this step, the navigation menu contains 4 elements: Plant, Animal, Bacteria and Archaea. From here, the algorithm is applied recursively until the menu is completely generated. In the next step, the Animal class is chosen and divided. However, in this case, there is not space for all its subclasses in the first level of the menu. For this reason, the subclasses with a higher number of instances move up to the main level of the menu while the rest of subclasses are grouped inside Other Animal.

It is important to highlight that the procedure depicted so far takes into account the whole dataset classes and instances at a given moment and generates the corresponding menu as a static snapshot. Figure 6.2 shows the complete menu generated for the DBpedia dataset.



Figure 6.2: Navigation menu generated for the DBpedia

6.1.2. Revising the algorithm

Although the generated navigation menus provide an overview of the dataset, we identified two important issues during our evaluation in Iteration 3 (section 7.3).

The main issue is related with the split operation, which divides a node into its subclasses when there is available space in the menu. This operation removes those nodes that have been completely divided, i.e. all their subclasses have been moved up a level in the hierarchy. However, this approach is only valid when all the instances of a node belong to one of its subclasses. Otherwise, the node should not be removed because some of their instances will not be accessible from navigation menus.

Let's consider a similar example as in the previous version. In this case, the number of instances of each class is different because we are using a newer version of DBpedia (v3.9). In our example we have the following classes hierarchically organized:

- Eukaryote: 235.699 resources.
 - Animal: 178.289 resources.
 - Plant: 47.175 resources.
 - Fungus: 8.739 resources.

As it can be seen, the number of resources that belong to Eukaryote is not the same as the sum of the resources of its subclasses. There are 199.085 resources of Eukaryote that do not belong to Animal, Plant or Fungus. In this case, the Eukaryote class should be splitted into:

- Animal: 178.289 resources.

6.1. OVERVIEW GENERATION AND STORAGE

- Plant: 47.175 resources.
- Fungus: 8.739 resources.
- Other Eukaryote: 1.496 resources.

The second issue is related with the factor considered to sort classes. Right now, the number of instances of each class is the only criterion used to select which class should be merged or expanded. While being important, it is not sufficient on its own and it does not ensure that the selected classes will be relevant in the dataset. A class may have a large number of instances but almost no properties at all, thus probably not describing relevant information. In other cases, a class at the top level may have no subclasses, wasting space in the navigation menu that could be used to show another class with subclasses.

Our proposal is to choose the most suitable classes to merge or expand according to a score function. Instead of considering only the number of instances of each class, we consider the following metrics, which all range from [0..1]:

- Instance frequency: computed as the number of instances of a class C in a dataset D . We normalize this value dividing it by the total number of instances n_i in the dataset.

$$I(C, D) = \frac{n_i(C)}{n_i}$$

- Number of subclasses: a class with subclasses is more suitable to be displayed in a hierarchical menu because it makes a better use of the space. We compute this metric as the number of subclasses $n_s(C)$ of a class C in a dataset D and then normalizing the value.

$$S(C, D) = \frac{n_s(C)}{\max\{\forall C_i \in D \rightarrow n_s(C_i)\}}$$

- Number of properties: the number of properties is a criteria to ensure that the selected classes are richly described. We compute this metric as the number of properties $n_p(C)$ of a class C in a dataset D . This value is then normalized.

$$P(C, D) = \frac{n_p(C)}{\max\{\forall C_i \in D \rightarrow n_p(C_i)\}}$$

The first two metrics are calculated with the SPARQL queries detailed in the previous section, Example 7 and Example 8, which retrieve the list of subclasses and number of instances of a class. The number of properties for a class is obtained with the SPARQL query from Example 11.

```
1 SELECT count(distinct(?p)) WHERE {  
2   ?r a <CLASS> .  
3   ?r ?p ?o  
4 }
```

Example 11: SPARQL query to count the number of properties for a <CLASS>

Finally, the score of a class C in a dataset D is calculated as a weighed multiplication:

$$Score(C, D) = W_I * I(C, O) + W_S * S(C, O) + W_P * P(C, O)$$

```

1 generateMenu(Menu menu, int numItems) {
2   menu.removeEmpty();
3   for(Node node : menu.getAllNodes()){
4     node.calculateScore();
5   }
6   while(menu.size()>numItems) {
7     Node other = menu.createOther();
8     Node min = menu.getMinNode();
9     other.mergeWith(min);
10  }
11  while(menu.size()<numItems){
12    Node max = menu.getMaxNode();
13    menu.splitNode(max);
14    if(max.numInstances==0)
15      menu.removeNode(max);
16  }
17 }

```

Example 12: Revised navigation menus algorithm

In our implementation we used $W_I = 0.6$, $W_S = 0.1$ and $W_P = 0.3$. As it can be seen, the number of instances is still the most important criterion. The number of subclasses and properties are used to adjust the score and try to ensure that the selected classes contain relevant information. The revised version of the algorithm, incorporating these changes, is shown in Example 12.

6.2. Facet discovery and ranking

Traditional facet browsers rely on manual identification of the facets and on a previous knowledge of the target domain. Facet browsers are developed to navigate through homogeneous data and facets are fixed. Domain experts need to be involved to decide manually which facets are relevant to the user. This conflicts with the Semantic Web, where data is too diverse to use a single set of facets. Since the Semantic Web integrates data from lot of sources, we can't assume a single fixed schema for all data: facets that make sense for one type of resources could be inappropriate for other types. Therefore, a semantic faceted browser should be able to handle any RDF dataset without any configuration. It should be scalable and generic, not depending on a particular dataset.

When dealing with semantic data, it is possible to automate this process. However, when a dataset is very large and heterogeneous, the number of facets will also be very large. Exposing all facets requires considerable screen size and can lead to excessive scrolling. Moreover, faceted browsing can potentially present too much information. It is important to prevent the "paradox of choice" [OHS09]. When users are presented with a large number of options, sometimes they make poor decisions.

This problem becomes particularly difficult in Semantic Web data, where large datasets can have hundreds of properties and thousands of values. Given the large datasets and the limited screen size, a crucial aspect of faceted navigation is to select the list of facets to display to the user.

6.2. FACET DISCOVERY AND RANKING

Let's consider the following example. A user wants to explore resources that belong to the class `http://dbpedia.org/ontology/Person` from the DBpedia. There are 763.644 resources belonging to this class and faceted browsing seems a good method to explore them and filter out those resources that are not interesting for the user. However, there are 11.596 distinct properties describing resources from this class. Which are the most common properties among all these resources? Which are the properties that best describe resources of the Person class?

Therefore, we need methods for ranking facets and select the most important facets for the user. We need to find those facets that best represent the dataset and those that are best to navigate the dataset, where the number of facets to select is usually a small number. A suitable facet should allow efficient navigation through the dataset and be representative for those objects.

6.2.1. Approaches to facet ranking

Automating facet discovery and ranking has been noted as a major research challenge, also in traditional systems and databases [Hea06b, DRM⁺08, DIW05]. In this section we summarize different approaches for ranking facets that can also be applied for exploring RDF data.

6.2.1.1. Frequency-based ranking

This technique ranks facets by the number of resources that have that property. Those properties with the larger number of resources are ranked higher. This heuristic allows users to see first facets with the greatest wealth of information. It also guarantees that low-ranked facets represent a small portion of the collection.

6.2.1.2. Set-cover ranking

The objective of this technique is to maximize the number of distinct resources that are accessible from the *top-k* ranked facets. This ranking tries to maximize the cardinality of the set $F_1 \cup F_2 \cup \dots \cup F_k$, where $F_1 \dots F_k$ are the *top-k* facets selected.

However, this problem is an instance of the set-cover problem, a NP-complete problem [CLRS09]. Consequently, the optimal ranking that selects the maximum number of resources from the collection, would take an exponential time to be calculated.

6.2.1.3. Metric-based ranking

Both algorithms presented above try to maximize the number of objects that are covered by the top-k facets selected, regardless of how useful are these facets. Oren et al. [ODD06] propose metrics to rank facets for the exploration of RDF data. The metric-based ranking method takes into consideration the frequency of each property but also the property balance, i.e. its entropy, and the object cardinality. As a result, this method ranks higher those facets that seem the best to navigate through the dataset.

6.2.2. Experimenting with metric-based ranking

The metric-based ranking proposed by Oren et al. [ODD06] is the most complete method to rank facets for RDF exploration. Therefore, it was our starting point for ranking facets in our faceted browser. In the following we describe the three metrics proposed by the authors and we also discuss some limitations of this approach, which motivated our proposal.

To measure the usefulness of a facet, and therefore showing it more prominently to the user, the metric-based ranking proposes three metrics:

- **Predicate frequency:** we are interested in those predicates that occur frequently inside the instances being browsed. The more resources covered by a predicate, the more useful it is in dividing the information space. If the predicate is not frequent it will only affect a small subset of the collection. We compute the predicate frequency as the number of resources for which the predicate p is defined. We normalise this value dividing it by the total number of resources:

$$freq(p) = \frac{n_r(p)}{n_r}$$

- **Predicate balance:** a facet helps the user better discriminate the set of instances being browsed when it takes a well-balanced range of values for the facet property. On the contrary, a facet whose property takes always or mainly a particular value is less useful. The same happens if each instance has a different value for the facet property. Consequently, we will favour facets that show behaviours in between these worst cases. To compute the predicate balance we use the Shannon's entropy formula:

$$H(S) = - \sum_{i=1}^n p(v_i) \log_n p(v_i)$$

- **Value cardinality:** a suitable predicate should have a small amount of values to choose from. If there are too many choices it is difficult to display all the options and it might confuse the user. We compute the value cardinality as the number of different values for a predicate. This metric is normalized using a function based on the Gaussian density that can be regulated through the μ and σ parameters to the top and bottom values of the range of different values we are interested in. The authors recommend ranges similar to from 2 to 20:

$$card(p) = \begin{cases} 0 & \text{if } n_o(p) \leq 1 \\ e^{-\frac{(n_o(p)-\mu)^2}{2\sigma^2}} & \text{otherwise} \end{cases}$$

Although this approach to rank facets is valid for small RDF datasets, when applied to large datasets it suffers from some limitations and scalability issues. First of all, the predicate balance metric, computed as Shannon's entropy, cannot be calculated in runtime. The SPARQL query to calculate this metric, shown in Example 13, uses the GROUP BY and COUNT functions, which require considerable time.

6.2. FACET DISCOVERY AND RANKING

```
1 SELECT ?o (COUNT(?o) AS ?n) ?label
2 WHERE {
3   ?r a <CLASS>; <PROPERTY> ?o .
4   OPTIONAL{ ?o rdfs:label ?label }
5 }
6 GROUP BY ?o ?label
7 ORDER BY DESC(?n)
```

Example 13: SPARQL query to obtain values and counts for a <CLASS> and <PROPERTY>

Secondly, the value cardinality metric does not make much sense in large datasets. They tend to have properties with a high number of different values, thus resulting in a low score on this metric. Thirdly, the authors suggest that facet metrics should be recalculated dynamically at each step, since the information space changes every time. Therefore, facets order should be updated accordingly and most of the SPARQL queries need to be executed several times. However, these changes in the UI could cause usability problems [NL06].

Finally, as already pointed out by the authors, these metrics are only an indication of the usefulness of a facet considering its structural properties. Although they divide the search space optimally, these selected facets do not necessarily correspond to those that best describe data according to users. Badly ranked facets could still be representative of the data and be intuitive for users.

6.2.3. Descriptive facet ranking

All the approaches previously described have several limitations. These issues and our experiments motivated the proposal of a new method to rank facets for the exploration of RDF data. Our objective is to devise a method to rank facets considering their usefulness but also their descriptive value for users.

Faceted browsing is based on the theory of facet analysis proposed by S.R. Ranganathan [Ran62] in the 1930s to improve bibliographic classification systems. According to Ranganathan, an intuitive facet should belong to either of these categories:

- **Personality:** the distinguishing characteristic of a subject, e.g. a person, animal, organization, etc.
- **Matter:** the physical material of a subject, e.g. a topic, colour, etc.
- **Energy:** an action that occurs to a subject, e.g. an activity, event, etc.
- **Space:** a geographic location of a subject, e.g. place of birth, event location, etc.
- **Time:** a period associated with a subject, e.g. year, date of birth, etc.

Ranganathan's theory could help us to automatically detect descriptive facets. A facet should only represent an important characteristic of the classified objects. Those facets that belong to these categories are likely to be descriptive of the data and intuitive for most users. A short path to this problem might be to identify those common properties and classes that usually represent these characteristics in RDF data. For example, we could say that the properties `foaf:name` or `vcard:fn` indicate a person, or the properties `wgs84:lat` and `wgs84:lon` represent a location.

However, this approach is only valid for some datasets, those that use these vocabularies, properties or classes. There may be datasets that use other vocabularies that also represent these categories. Our goal is to devise a method to rank facets without relying on concrete vocabularies and properties.

6.2.3.1. Metrics proposed

In this section we describe the metrics proposed to rank facets according to their descriptive value of the data. We also include the property frequency to ensure that the selected facets are also useful for users to explore data.s

- **Defined in ontology:** ontologies can provide us metadata to identify facets that are descriptive of a class, probably belonging to the categories proposed by Ranganathan. It seems appropriate to consider those properties defined in the ontology as relevant. We compute this metric using the following function:

$$O(p) = \begin{cases} 0 & \text{if the property } p \text{ is not defined in ontologies} \\ 1 & \text{if the property } p \text{ is defined in ontologies} \end{cases}$$

- **Descriptive range:** the introduction of pivoting (in iteration 2, section 7.2) and facet widgets (in iteration 5, section 7.5), made us consider this metric. Those properties with a specific range can also have a descriptive value. Concretely, we consider the following properties to have a descriptive value:
 - Properties representing ordinal data, e.g. numbers, dates, time, etc. Their range belong to these data types from XML Schema¹³: `xsd:int`, `xsd:decimal`, `xsd:float`, `xsd:double`, `xsd:date`.
 - Object properties, whose values link to other resources. These properties allow pivoting, as detailed in section 7.2.2.1. These properties have also a descriptive value, since they are linked and related to other resources. In fact, this is one of the four principles of Linked Data: to include links to related things.

We compute this metric, descriptive range $R(p)$, using the following function, being L the list of ranges considered to have a descriptive value:

$$R(p) = \begin{cases} 0 & \text{iff } range(p) \notin L \\ 1 & \text{iff } range(p) \in L \text{ or } range(p) \text{ results in } pivoting. \end{cases}$$

- **Pivoting score:** the importance of those properties that result in pivoting depends on the importance of the classes they pivot to. The more important the pivoted class is, the more important will be the property. Therefore, we compute the pivoting score $S(p)$ of a property p as the score calculated for the class c it pivots to.

$$S(p) = \begin{cases} 0 & \text{iff } range(p) \text{ does not result in pivoting} \\ score(c) & \text{iff } range(p) \text{ results in pivoting to class } c \end{cases}$$

¹³<http://www.w3.org/TR/xmlschema11-2/#built-in-datatypes>

6.2. FACET DISCOVERY AND RANKING

The score of a class c is calculated with the function described in subsection 6.1.2, taking into account the number of instances, properties and subclasses of the class.

- **Property frequency:** the selected facets should be statistically significant in the dataset, being present in as many resources as possible. The property frequency is computed as the number of resources n_r that have the property p , divided by the total number of resources n_r :

$$F(p) = \frac{n_r(p)}{n_r}$$

All metrics range from $[0..1]$ and they are combined into a final score through weighted multiplication that produces a unique usefulness value for each facet:

$$Rank(p) = W_F * F(p) + W_O * O(p) + W_R * R(p) + W_P * P(p)$$

In our experiments we used $W_F = 0.6$, $W_O = 0.1$, $W_R = 0.1$ and $W_P = 0.2$. We gave the property frequency a higher coefficient because we believe it should be considered as the most important metric. A facet can be the most intuitive for users, but it can be useless if only a few resources have that property.

In addition to this ranking, it is also possible to define a list of properties to be always included or omitted in the list of facets. The site administrator can define the following parameters:

- **Whitelist:** a list of properties that will be ranked higher and always included. It can be used to define common properties that should appear first or important properties that are ranked with a low score, which otherwise would not be included. These properties are automatically ranked the maximum value, i.e. 1.
- **Blacklist:** a list of properties that will be omitted. It can be used to skip properties with a high frequency that are useless for users to filter results, e.g. autonumeric ids or built-in properties from OWL, RDFS, etc. The properties from this list are automatically ranked the lowest value, i.e. 0.

Table 6.1 shows the ranking for the class `http://dbpedia.org/ontology/Ship`. The table presents the metrics used to calculate the final score for each property. Some prefixes were used to shorten URIs¹⁴ ¹⁵ ¹⁶ ¹⁷. In this case, the properties `rdfs:label` and `rdf:type` were included in the whitelist, being ranked the highest.

¹⁴dbo: <http://dbpedia.org/ontology/>

¹⁵xsd: <http://www.w3.org/2001/XMLSchema#>

¹⁶rdfs: <http://www.w3.org/2000/01/rdf-schema#>

¹⁷rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

| Property | Score | Inverse | Defined in Ontology | Descriptive Range | Range | Pivoting Score | Frequency |
|--|---------|---------|---------------------|-------------------|--------------------|----------------|-----------|
| http://www.w3.org/2000/01/rdf-schema#label | 1.0 | false | false | false | xsd:string | 0.0 | 0.466927 |
| http://www.w3.org/1999/02/22-rdf-syntax-ns#type | 1.0 | false | false | false | rdfs:Resource | 0.0 | 1.0 |
| http://dbpedia.org/ontology/wikiPageWikiLink | 0.44271 | false | false | true | dbo:Place | 0.351 | 0.45417 |
| http://dbpedia.org/property/shipFate | 0.41391 | false | false | true | dbo:Agent | 0.549 | 0.34017 |
| http://dbpedia.org/property/shipPropulsion | 0.41179 | false | false | true | dbo:Agent | 0.549 | 0.33663 |
| http://dbpedia.org/ontology/shipLaunch | 0.36812 | false | true | true | xsd:date | 0.0 | 0.2802 |
| http://dbpedia.org/ontology/shipBeam | 0.34666 | false | true | false | xsd:double | 0.0 | 0.41111 |
| http://dbpedia.org/ontology/commissioningDate | 0.34017 | false | true | true | xsd:date | 0.0 | 0.23361 |
| http://dbpedia.org/ontology/layingDown | 0.31271 | false | true | true | xsd:date | 0.0 | 0.18784 |
| http://dbpedia.org/property/shipLaidDown | 0.30803 | false | false | true | dbo:PopulatedPlace | 0.2912 | 0.24965 |
| http://dbpedia.org/property/shipArmament | 0.30364 | false | false | true | dbo:Weapon | 0.0135 | 0.33488 |
| http://dbpedia.org/ontology/decommissioningDate | 0.28839 | false | true | true | xsd:date | 0.0 | 0.14732 |
| http://xmlns.com/foaf/0.1/primaryTopic | 0.28156 | true | false | false | rdfs:Resource | 0.0 | 0.46927 |
| http://xmlns.com/foaf/0.1/isPrimaryTopicOf | 0.28156 | false | false | false | rdfs:Resource | 0.0 | 0.46927 |
| http://www.w3.org/ns/prov#wasDerivedFrom | 0.28156 | false | false | false | rdfs:Resource | 0.0 | 0.46927 |
| http://www.w3.org/2000/01/rdf-schema#comment | 0.28156 | false | false | false | xsd:string | 0.0 | 0.46927 |
| http://dbpedia.org/ontology/wikiPageRevisionID | 0.28156 | false | false | false | xsd:integer | 0.0 | 0.46927 |
| http://dbpedia.org/ontology/wikiPageID | 0.28156 | false | false | false | xsd:integer | 0.0 | 0.46927 |
| http://dbpedia.org/ontology/abstract | 0.28156 | false | false | false | xsd:string | 0.0 | 0.46927 |
| http://dbpedia.org/property/wikiPageUsesTemplate | 0.28149 | false | false | false | rdfs:Resource | 0.0 | 0.46916 |
| http://purl.org/dc/terms/subject | 0.28147 | false | false | false | rdfs:Resource | 0.0 | 0.46912 |
| http://dbpedia.org/ontology/homeport | 0.27022 | false | true | true | dbo:Place | 0.351 | 2.0E-5 |
| http://dbpedia.org/ontology/length | 0.25285 | false | false | false | xsd:double | 0.0 | 0.42141 |
| http://dbpedia.org/ontology/acquirementDate | 0.25213 | false | true | true | xsd:date | 0.0 | 0.08688 |
| http://dbpedia.org/ontology/status | 0.24832 | false | false | false | xsd:string | 0.0 | 0.41386 |
| http://xmlns.com/foaf/0.1/name | 0.23831 | false | false | false | xsd:string | 0.0 | 0.39718 |
| http://dbpedia.org/property/shipPower | 0.23638 | false | false | true | dbo:Agent | 0.549 | 0.04428 |

Table 6.1: Automatic facet ranking for the class `http://dbpedia.org/ontology/Ship`

```

1 SELECT DISTINCT ?p ?label ?range
2 WHERE {
3   ?x a <CLASS> ; ?p ?o
4   OPTIONAL { ?p rdfs:range ?range }
5   OPTIONAL { ?p rdfs:label ?label }
6   FILTER (?o != "")
7   FILTER (?p!=owl:differentFrom && ?p!=owl:sameAs)
8 }

```

Example 14: SPARQL query to obtain all properties for a `<CLASS>`

Example 14 shows the SPARQL query to obtain all the properties used to describe instances of that class in the dataset, and Example 15 shows the SPARQL query that obtains all properties defined in ontologies for that class. These queries also retrieve their range and labels when they are available. When the property range is not defined, it is determined from the 5 most common values of that property. This process is described in more detail in iteration 2 (section 7.2) with the introduction of pivoting. Finally, Example 16 shows the SPARQL query to obtain the number of resources that have a concrete property, necessary to calculate the property frequency. These queries are used to retrieve all the information necessary to build facets and calculate the previous metrics. They are performed for each class in the dataset and are stored in a local `sqlite`¹⁸ database.

¹⁸<http://www.sqlite.org/>

6.2. FACET DISCOVERY AND RANKING

```
1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX owl: <http://www.w3.org/2002/07/owl#>
4 PREFIX co: <http://rhizomik.net/ontologies/copyrightonto.owl#>
5 PREFIX schema: <http://schema.org/>
6
7 SELECT DISTINCT ?p ?label ?r ?rlabel WHERE {
8   { ?p a ?type . FILTER(?type=owl:ObjectProperty || ?type=owl:DatatypeProperty) .
9     ?p rdfs:domain <CLASS> .
10    OPTIONAL { ?p rdfs:label ?label }
11  }
12 UNION
13   { ?restr rdf:type owl:Restriction; owl:onProperty ?p.
14     <CLASS> rdfs:subClassOf ?restr.
15     ?restr owl:allValuesFrom ?r.
16     OPTIONAL { ?p rdfs:label ?label }
17     OPTIONAL { ?r rdfs:label ?rlabel }
18   }
19 UNION
20   { ?restr rdf:type owl:Restriction; owl:onProperty ?p.
21     <CLASS> rdfs:subClassOf ?restr.
22     ?restr owl:someValuesFrom ?r.
23     OPTIONAL { ?p rdfs:label ?label }
24     OPTIONAL { ?r rdfs:label ?rlabel }
25   }
26 UNION
27   { ?restr rdf:type owl:Restriction; owl:onProperty ?p.
28     <CLASS> rdfs:subClassOf ?restr.
29     ?restr owl:hasValue ?r.
30     OPTIONAL { ?p rdfs:label ?label }
31     OPTIONAL { ?r rdfs:label ?rlabel }
32   }
33 }
```

Example 15: SPARQL query to obtain ontology properties for a <CLASS>

```
1 SELECT (COUNT(DISTINCT(?r)) AS ?n)
2 WHERE {
3   ?r a <CLASS> ; <PROPERTY> ?o
4 }
```

Example 16: SPARQL query to obtain the number of instances of a <CLASS> that have a concrete <PROPERTY>

These pre-computed facets are used whenever a user wants to browse a class, usually selecting it from the navigation bar. When a user starts interacting with them by selecting values, facets are dynamically recalculated starting from the pre-computed ones. This makes it possible to provide reasonable response times to users because most of the computing effort has already been done. The user interaction only restricts the set of resources to work with. The set of instances used for facets generation is constrained by the choices made so far and the facets are recalculated for that subset.

Once facets have been generated and selected given their usefulness, we keep their order in the user interface because too many changes in the UI could cause usability problems [NL06]. Despite some facets may contain no active values after applying some filters, it is important to keep displaying it in order to retain consistency in the interface. *Rhizomer* can also order facets by their name. Some studies suggest that in some cases users prefer a natural ordering [PH99]. If these facets that are automatically suggested are not enough, the user can add more filters using a button.

6.3. Linked Data Visualization Model

Applying information visualization techniques to the Semantic Web helps users to explore large amounts of data and interact with them. The main objectives of information visualization are to transform and present data into a visual representation, in such a way that users can obtain a better understanding of the data [CMS99]. Visualizations are useful for obtaining details of data and comprehend it. They can also help to get an overview of the datasets, their main classes, properties and the relationships between them.

However, the large amount of Linked Data available and its heterogeneity makes it difficult to find the right visualization for a given dataset, being not an easy task [HBO]: *‘One must determine which questions to ask, identify the appropriate data, and select effective visual encodings to map data values to graphical features such as position, size, shape, and color. The challenge is that for any given data set the number of visual encodings – and thus the space of possible visualization designs – is extremely large.’*

Compared to prior information visualization strategies, we have a unique opportunity on the Data Web. The unified RDF data model being prevalent on the Data Web enables us to bind data to visualizations in an unforeseen and dynamic way. An information visualization technique requires certain data structures to be present. When we can derive and generate these data structures automatically from reused vocabularies or semantic representations, we are able to realize a largely automatic visualization workflow. Ultimately, we aim to realize an ecosystem of data extractions and visualizations, which can be bound together in a dynamic and unforeseen way. This will enable users to explore datasets even if the publisher of the data does not provide any exploration or visualization means.

6.3.1. Overview of LDVM

Chi’s *Data State Reference Model* [Chi00] defines the visualization process in a generic way. It describes a process for transforming raw data into a concrete visualization by defining four data stages as well as a set of data transformations and operators. This framework provides a conceptual model that allows to identify all the components in the visualization process.

We use the *Data State Reference Model* (DSRM) proposed by Chi as conceptual framework for our *Linked Data Visualization Model* (LDVM). While the DSRM describes the visualization process in a generic way, we instantiate and adopt this model with LDVM for the visualization of RDF and Linked Data. The main difference is that in certain parts, LDVM works solely with RDF data model for increased automation. The names of the stages, transformations and operators have been adapted to the context of Linked Data and RDF. Figure 6.3 shows an overview of LDVM.

LDVM resembles a pipeline starting with raw source data and results with a visualization of the source data. It is organized into four stages that source data needs to pass through:

1. **RDF Data:** the raw data, which can be all kinds of information adhering to the RDF data model, e.g. instance data, taxonomies, vocabularies, ontologies, etc.
2. **Analytical extraction:** data extractions obtained from raw data, e.g. filtering, calculating aggregated values.

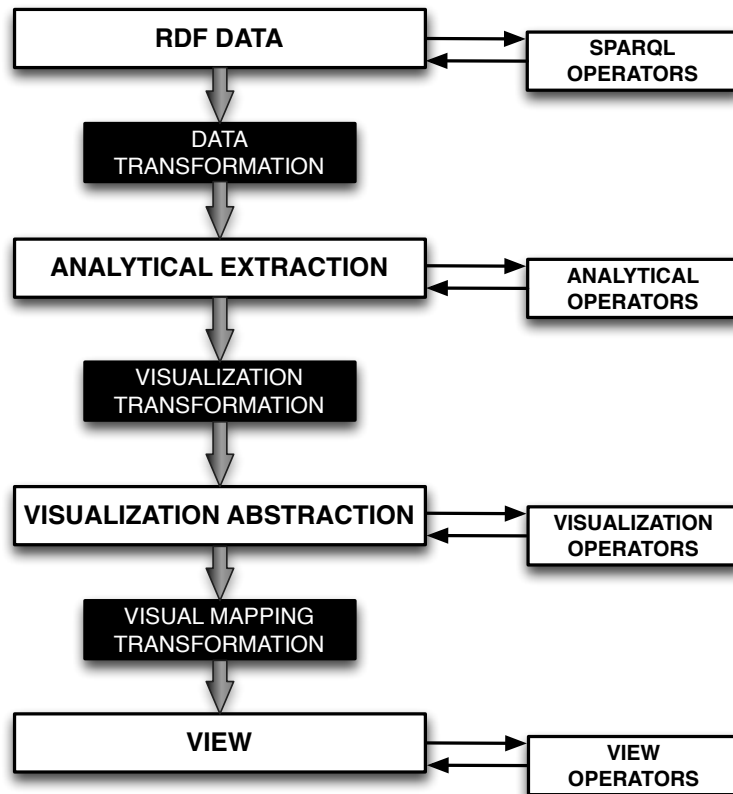


Figure 6.3: High level overview of the Linked Data Visualization Model.

3. **Visual abstraction:** preparation of an RDF data structure required by a particular visualization technique; the data structure is based on generic data types for visual analysis proposed by Shneiderman [Shn96] (i.e., 1D, 2D, 3D or multi-dimensional data, temporal data, tree data, or network data)
4. **View:** the result of the process presented to the user, e.g. plot, treemap, map, timeline, etc.

Data is propagated through the LDVM pipeline from one stage to another by applying three types of transformation operators:

1. **Data transformation:** transforms raw data values into analytical extractions declaratively, using SPARQL queries.
2. **Visualization transformation:** takes analytical extractions and transforms them into a visualization abstraction.
3. **Visual mapping transformation:** maps the visualization abstraction data structure to a concrete visual structure on the screen using a particular visualization technique.

There are also operators within each stage that do not change the underlying data but allow to extract data from each stage or add new information. These are:

1. **SPARQL Operators:** SPARQL functions, e.g. SELECT, FILTER, COUNT, GROUP BY, etc.
2. **Analytical Operators:** further refine the data extractions, e.g. select a subset of data.
3. **Visualization Operators:** visual variables, e.g. visualization technique, sizes, colors, etc.
4. **View Operators:** actions applied to the view, e.g. rotate, scale, zoom, etc.

6.3.2. LDVM Stages

In the first stage we consider RDF data, but non-RDF data sources could also be considered. Since data extraction is a vast research field on its own, we do not consider non-RDF data sources and we focus on RDF data for simplicity reasons. A survey on knowledge extraction approaches can be found in [UHAS12]. The transformation can be one or more SPARQL queries which map the source data to the required structure applying filters and other operators. In the case of non-RDF data sources, this transformation should also convert the source data model to the RDF data model, e.g. CSV to RDF.

The output of the second stage (analytical extraction) is produced by applying a sequence of various in-stage SPARQL operators on the RDF data. The user can create analytical extractions by applying filters and reducing the data to a desired level. In this stage, we also want to provide users analytical extractions that can be applied to a given dataset. Therefore, it will be necessary to be able to decide whether or not an extraction can be applied on a given dataset, i.e. determinate whether or not they are compatible.

An analytical extraction is not a suitable data structure for visualization. A visualization tool displays particular generic characteristics captured by the analytical extractions. Therefore, we need to transform the analytical extraction into a more generic data structure based on Generic Visualization Data Types (GVDTs), listed in Table 6.2. The GVDTs are inspired by the data types proposed by Shneiderman [Shn96]. This structure is then what is visualized by visualization tools.

In order to facilitate associating visualization tools to analytical extractions, we provide mappings from GVDTs to visualization tools but also to RDF vocabularies. In this way, it is possible to associate input analytical RDF extractions to GVDTs and then provide or recommend the more suitable visualization tools to deal with them. The mappings are summarized in Table 6.2. For example, geospatial data modeled using the `wgs84:lat` and `wgs84:lon` properties, can be visualized by any visualization tool to render maps, e.g. Google Maps, OpenStreetMap, etc.

Finally, the output of the view stage is a visual representation of a visualization abstraction on the screen. It may be configured by a user using various parameters, e.g. visualization technique, colors and shapes. The user can also manipulate the final view using the view in-stage operators such as zoom or move.

6.3. LINKED DATA VISUALIZATION MODEL

| RDF Vocabulary | Data Type | Visualization Tool |
|------------------------------------|------------------|------------------------|
| xsd:int, dc:subject,... (count) | 1D | Histogram |
| wgs84:lat, geo:point,... | 2D | Map |
| visko:3DPointPlot,... | 3D | 3D Rendering |
| qb:Observation, scovo:Item,... | Multidimensional | Chart |
| xsd:date, ical:dtstart,... | Temporal | Timeline, Calendar,... |
| rdfs:subClassOf, skos:narrower,... | Tree | Treemap, SunBurst,... |
| foaf:knows,... | Network | Graph,... |

Table 6.2: Generic visualization data types.

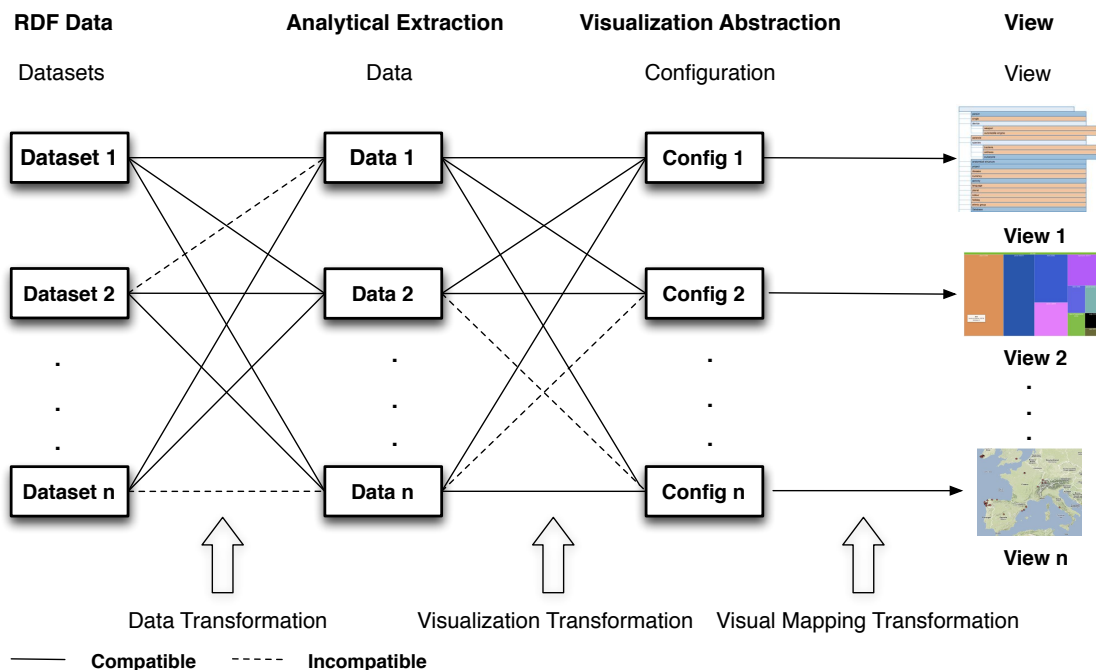


Figure 6.4: Linked Data Visualization Model ecosystem

As illustrated in Figure 6.4, our model allows to connect different RDF datasets and different data extractions with different visualization techniques. Not all datasets are compatible with all data extractions and each data extraction is only compatible with some visual configurations. Each dataset offers different data structures to be extracted, e.g. charts for numerical values, a class hierarchy, geospatial data, etc. Each data extraction can be visualized with different configurations, which contain information such as the visualization technique to use, colors, etc. Then, a concrete visualization is generated depending on the data extraction and the visual configuration.

To summarize, the model is divided into two main areas: data space and visual space. The RDF data stage, analytical extraction stage and data transformation belong to the data space, while visual abstraction stage, view stage and visual mapping transformation belong to the visual space. These two main blocks are connected by a visualization transformation.

6.3.3. Formalization and compatibility

The RDF data model enables us to bind data to visualizations in a dynamic way while the LDVM allows to connect datasets with data extractions and visualizations. However, the compatibility between data and concrete visualizations needs to be determined.

In this section we formalize the extraction and generation of visualization data structures (in the analytical extraction stage) as well as the visualization configuration (in the visualization abstraction stage). These structures capture the information required to:

- Process a dataset and extract a relevant subset (or aggregation) of the data to visualize.
- Determine whether or not an analytical extraction is compatible with a certain visualization.

Definition 1 (Visualization Data Extraction) *The visualization data extraction defines how data is accessed for the purpose of visualizations. Our visualization data extraction structure comprises the following elements:*

- a possibly empty set D of data extraction parameters,
- a set tuples (q, σ_q) , with q being a SPARQL query template containing placeholders mapped to the data extraction parameters D and σ_q being the signature of the SPARQL query result, i.e. the number of columns returned by queries generated from q and their data types.

Visual data extractions define how data can be extracted declaratively using SPARQL query templates for visualization. Visual configurations are the corresponding structures on the visualization space side, that can receive extracted data:

Definition 2 (Visualization Configuration) *The visualization configuration is a data structure, which accommodates all configuration information required for a particular type of visualization. This includes:*

- the visualization technique, e.g. Google Maps, Histogram, Timeline, Treemap, etc,
- the visualization data type from Table 6.2, e.g. 1D, 2D, Temporal, Tree, etc.
- visualization parameters, e.g. colors, sizes,

The declarative definition of the data extraction and visualization configuration enables flexible combination of extractions and configurations. However, we have to ensure, that the structure of extracted data is compatible with the structure of data required for a visualization. For that purpose, we define the compatibility between visualization data extraction and visualization configuration based on matching signatures:

Definition 3 (Compatibility) *An input signature S is a pair (VC, Q) where V is a Visualization Configuration and Q is a set of SPARQL queries. A visualization data extraction VDE is said to be compatible with a visualization configuration VC iff each q in Q returns a non-empty result when executed on VDE .*

The compatibility is checked normally using a SPARQL ASK query, which determines whether a certain dataset contains relevant information for this visualization.

6.3.4. Implementation

Based on LDVM, we implemented a comprehensive two-level prototype. *LODVisualization*¹⁹ implements our architecture on the overview level. It allows users to explore and view the Data web through different visualizations on the overview level of detail. These visualizations allow users to obtain an overview of RDF datasets and realize what the data is about: their main types, properties, etc. Secondly, we have also implemented the LDVM in *Rhizomer*. In this case, the implementation focuses on the details on demand part of Shneiderman's mantra [Shn96], providing visualizations of specific data.

6.3.4.1. LODVisualization

LODVisualization is a prototype implementing the LDVM that allows users to explore and interact with the Data Web through different visualizations. It allows to connect different datasets, different data extractions and different visualizations according to the LDVM in a dynamic way. In this way, our prototype serves not only as a proof-of-concept of our LDVM but also provides useful visualizations of RDF. These visualizations allow users to obtain an overview of RDF datasets and realize what the data is about: their main types, properties, etc.

In *LODVisualization* users can enter or select a SPARQL endpoint and select the graphs to visualize (first step of LDVM model). Then, the compatibility between data extractions and datasets is checked in order to determine which of them are available (second step). Once the data extraction has been executed, the results are stored into a visual abstraction, which corresponds to the third step of the model. Finally, users can visualize the results using different visualizers depending on their compatibility (fourth step).

LODVisualization is developed using *Google App Engine* (GAE), a cloud computing platform for developing and hosting web applications on Google's infrastructure. Figure 6.5 shows the architecture of *LODVisualization*.

The frontend is developed using HTML, CSS and Javascript, while the backend is developed in Python. Most of the visualizations are created using the *JavaScript Infovis Toolkit*²⁰ and *D3.js* [BOH11]. We have chosen them because they allow to create rich and interactive JavaScript visualizations and they also include a free license. We also use *Google Maps* and *OpenStreetMaps* to create geospatial visualizations. Data is transferred between SPARQL endpoints, the server and the client using JSON, which is the data format used by those libraries. Most SPARQL endpoints support JSON and can convert RDF to JSON, thus facilitating the integration with Javascript.

¹⁹lodvisualization.appspot.com

²⁰<http://thejit.org/>

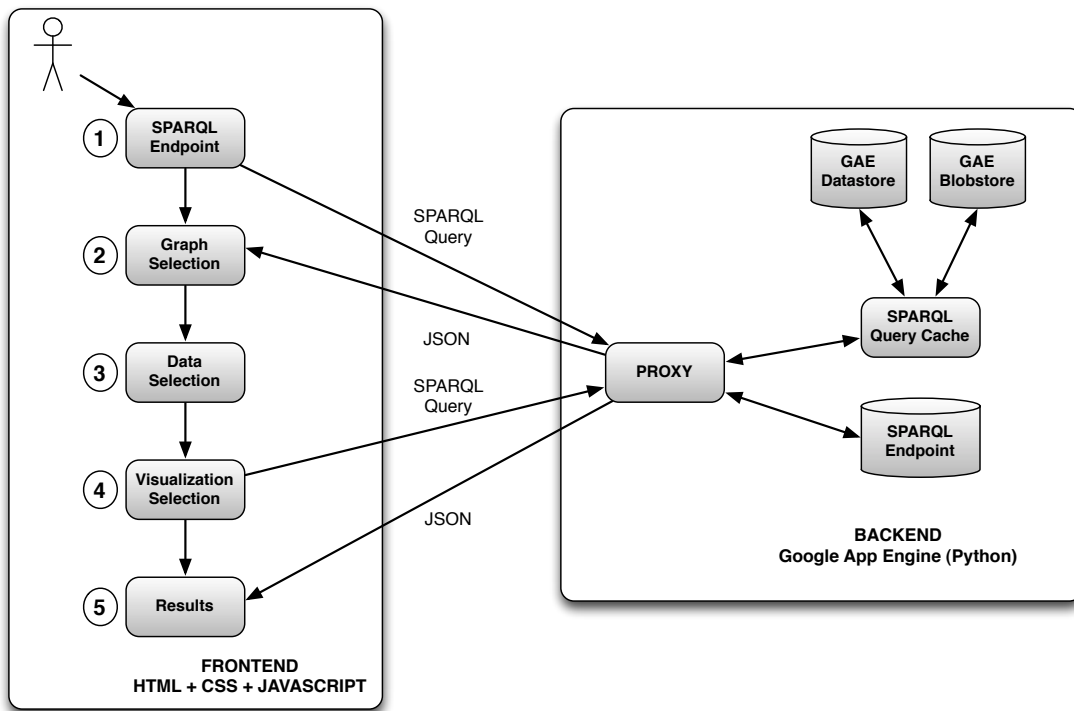


Figure 6.5: High-level LODVisualization architecture.

Instead of querying SPARQL endpoints directly using AJAX, queries are generated on the client side but sent to the server. The server acts as a proxy in order to avoid AJAX cross-domain requests. At the same time, the server provides a cache infrastructure implemented using GAE Datastore and GAE Blobstore. For each SPARQL query we generate an MD5 hash and store the hash along with the query results. The GAE Blobstore is used to save those results whose size exceeds 1MB since the Datastore has this limitation. When the server receives a query, the results are obtained from cache if available and not stale. Otherwise, the query is sent to the SPARQL endpoint using the GAE URL Fetch API. Then, the hash of the query and its results are stored in the cache. The cache substantially increases performance, scalability and reliability of Linked Data visualizations by preventing from executing the same queries when users aim to visualize the same data extraction with different visualization techniques.

LODVisualization is compatible with most of SPARQL endpoints as long as JSON and SPARQL 1.1²¹ are supported. Most of the data extraction queries use aggregate functions such as COUNT or GROUP BY, which are implemented starting from this version. Our implementation has a limitation: the response of each SPARQL query must be available in less than 60 seconds. This is the maximum timeout for requests using the GAE URL Fetch API. Nevertheless, longer requests could be performed using the GAE Task Queue API or an alternative infrastructure.

²¹<http://www.w3.org/TR/sparql11-query/>

6.3. LINKED DATA VISUALIZATION MODEL

Our implementation includes data extractions such as the class hierarchy, property hierarchy, SKOS concepts hierarchy, properties connecting two classes, etc. The results can be visualized using techniques such as tables, treemaps, charts, maps, etc. Since being based on the LDVM, *LODVisualization* is easy to extend with additional data extractions and visualization techniques.

6.3.4.2. Rhizomer

The LDVM has also been implemented in *Rhizomer*. In this case, it is integrated with existing visualizations for details on demand tasks. *Rhizomer* can be deployed on top of any SPARQL endpoint and configured to explore a RDF graph. When users interact with the system, they can create their own extractions by combining filters in the faceted browser. Once data has been reduced to a set of items of interest, it can be visualized using different techniques when they are compatible.

The preliminary version of *Rhizomer* already includes some visualizations such as map, timeline and charts. These visualizations are not a contribution of these work and more details about their implementation can be found in [Muz09]. However, they have been integrated following the LDVM. They are available when they are compatible with data extractions.

For example, the map visualization is available when a data extraction contains properties that represent geospatial points, i.e. `wgs84:lat`, `wgs84:lon`, `georss:point`, `vcard:latitude` or `vcard:longitude`. The compatibility is determined with the SPARQL ASK shown in Example 17.

```
1 PREFIX georss: <http://www.georss.org/georss#>
2 PREFIX wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#>
3 PREFIX vcard: <http://www.w3.org/2006/vcard/ns#>
4 ASK WHERE {
5   { ?r georss:point ?point .
6     FILTER(?r =<[URI]>) }
7   UNION
8   { ?r wgs84_pos:lat ?lat; wgs84_pos:long ?long .
9     FILTER(?r =<[URI]>) }
10  UNION
11  { ?r vcard:geo ?geo. ?geo vcard:latitude ?lat; vcard:longitude ?long .
12    FILTER(?r =<[URI]>) }
13 }
```

Example 17: SPARQL query to determine the compatibility of map visualizations

This query returns a boolean indicating whether the pattern matches or not. A similar query is executed for the other visualizations in order to determine if data extractions contain dates to display on a timeline or numerical values to display on a chart. Figures 6.6 and 6.7 show the map and chart visualizations.

CHAPTER 6. AUTOMATIC INFORMATION ARCHITECTURE GENERATION METHODS

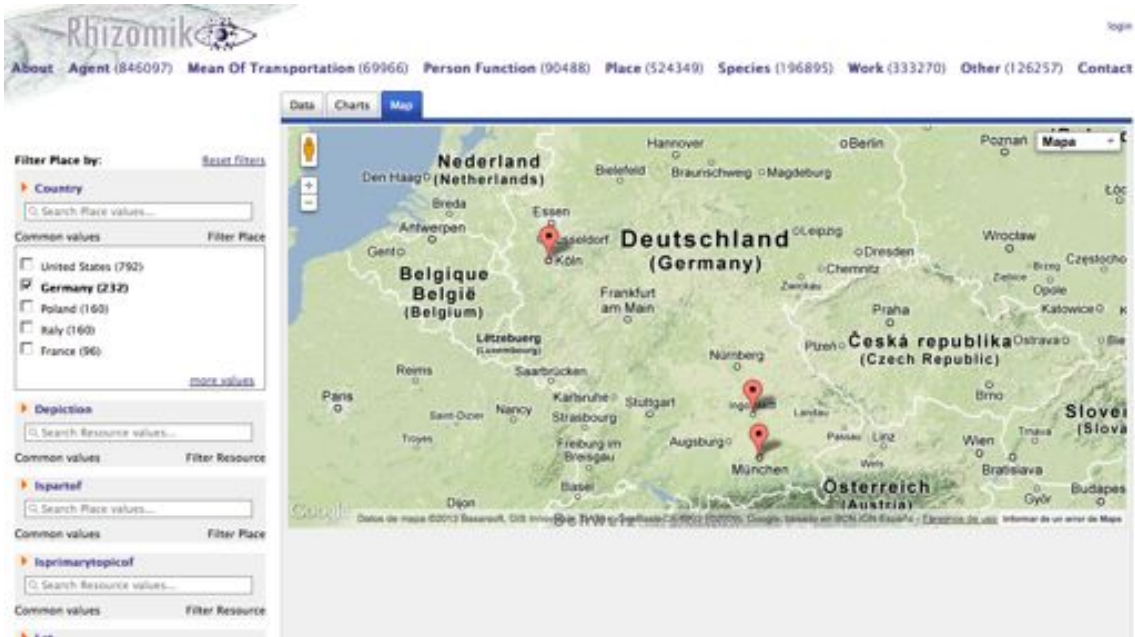


Figure 6.6: Map visualization.



Figure 6.7: Chart visualization.

6.3.5. Evaluation

The goal of our evaluation was to prove that the LDVM can be applied to different datasets providing different data visualizations. In order to demonstrate the feasibility to perform visualizations, we evaluated *LODVisualization* with different datasets, data extractions and visualizations. The evaluation was performed with Google Chrome version 19.0.1084.46. Table 6.3 shows a summary of the evaluation results.

| ID | Dataset | Data Extraction | Visualization Configuration | Execution time (s) | |
|----|---------------|------------------------|-----------------------------|--------------------|----------|
| | | | | w/o cache | w/ cache |
| 1 | DBpedia | Class hierarchy | Treemap | 3.58 | 0.87 |
| 2 | DBpedia | Class hierarchy | Crop Circles | 3.35 | 0.76 |
| 3 | DBpedia | SKOS concept hierarchy | Treemap | 48.79 | 16.96 |
| 4 | DBpedia | Spatial data | Google Maps | 2.05 | 0.37 |
| 5 | Dbpedia | Spatial data | OpenStreetMaps | 2.03 | 0.79 |
| 6 | Wine Ontology | Property hierarchy | Indented Tree | 1.95 | 0.57 |
| 7 | Wine Ontology | Property hierarchy | Space Tree | 2.45 | 0.70 |
| 8 | LinkedMDB | Class hierarchy | Treemap | 3.28 | 0.61 |
| 9 | WWW 2011 | Class hierarchy | Crop Circles | 2.53 | 0.75 |
| 10 | AGRIS – FAO | SKOS concept hierarchy | Treemap | 28.72 | 8.54 |

Table 6.3: Evaluation results summary: execution time for 10 combinations of datasets, data extractions and visualization configurations.

Timings for each concrete visualization were averaged from 10 execution cycles without cache and 10 execution cycles with cache. Despite using some of the largest datasets available on the Data Web, most of the visualizations can be generated in real-time (<5s rendering time) and the use of the cache further reduces the execution time substantially (<1s rendering time). However, in experiments #3 and #10 (SKOS concept hierarchies) the results had to be stored in the GAE Blobstore instead of the GAE Datastore due to their size. Accessing the Blobstore to retrieve those results is much slower than accessing the Datastore, but further optimizations are possible to increase performance in such cases.

Table 6.4 shows the timings for each visualization divided into the three transformations proposed in the LDVM: data transformation, visual transformation and visual mapping transformation.

Creating different visualizations for the same data extraction takes a similar time. This is due to the fact that most of the execution time can be attributed to the data transformation. The visual transformation timing depends on the size of the results to process. In the same way, the visual mapping transformation depends on the number of items to visualize. This number is particularly high in experiments #3 and #10, with a huge hierarchy of SKOS concepts.

| ID | Data Transformation - SPARQL Query Templates (s) | | | | Visual Transformation (s) | Visual Mapping Transformation (s) |
|----|---|--------|---------|---------|------------------------------|--------------------------------------|
| | #1 | #2 | #3 | Total | | |
| 1 | 1.4336 | 1.2741 | 0.7305 | 3.4382 | 0.0190 | 0.1228 |
| 2 | 1.2194 | 1.3423 | 0.7123 | 3.2740 | 0.0140 | 0.0620 |
| 3 | 3.5202 | 5.5424 | 35.9858 | 45.0484 | 0.4588 | 3.2900 |
| 4 | 2.0028 | - | - | 2.0028 | 0.0148 | 0.0356 |
| 5 | 1.6008 | - | - | 1.6008 | 0.0100 | 0.4266 |
| 6 | 0.7792 | 0.4860 | 0.6712 | 1.9364 | 0.0022 | 0.0130 |
| 7 | 0.98875 | 0.6757 | 0.5650 | 2.2295 | 0.0015 | 0.0372 |
| 8 | 0.9554 | 0.7334 | 1.5442 | 3.233 | 0.0086 | 0.0482 |
| 9 | 0.8914 | 0.8924 | 0.7016 | 2.4854 | 0.0102 | 0.0430 |
| 10 | 6.2942 | 6.4924 | 14.1782 | 26.9648 | 0.5428 | 1.2184 |

Table 6.4: Timing for each transformation: data transformation, visual transformation and visual mapping transformation.

However, it is important to highlight that the execution times are not really relevant for this evaluation. They depend mainly on the complexity of the SPARQL queries required for the data extraction as well as on the availability of SPARQL endpoints or Google App Engine servers. All these visualization examples are available on the website and it is easy to create new ones.

Iterative User Interface Development

In this chapter we describe the process of designing and implementing the user interface components. It is composed of 5 different iterations based on the RITE method [MWT⁺02]. Each iteration follows the MPIu+a process model [Gra03], including requirements analysis, design, implementation and evaluation steps. In each iteration, an evaluation of the quality in use was performed based on SWET-QUM. In the following, we summarize the main features of each iteration.

In the first iteration we identified the initial requirements for our system and implemented a basic set of components to support them: navigation menus, facets, breadcrumbs and an HTML view of resources to obtain details on demand.

In the second iteration we implemented pivoting in facets, allowing users to switch between different types of resources and create complex queries. Moreover, we introduced literal breadcrumbs in order to improve user contextualization during the exploration.

In the third iteration we implemented and evaluated three overview components that complement navigation menus and provide a more detailed overview: a site map, a site index and a treemap.

In the fourth iteration we redesigned the user interface components in order to encourage pivoting and reduce the number of interaction steps necessary to perform tasks. We also implemented path and hierarchical breadcrumbs to improve contextualization between pivoting steps.

Finally, in the fifth iteration we propose a search component and advanced widgets for facets. This iteration is still under development and has not been evaluated yet. It can be considered as future work.

7.1. Iteration 1

7.1.1. Requirements analysis

Based on the analysis of existing tools and their limitations, we derived a set of requirements for our system. These requirements are related with the challenges presented in subsection 1.2.2 and with the tasks for data analysis proposed by Shneiderman [Shn96]. A brainstorming session was also performed by some of the GRIHO members in order to get ideas and confirm the identified requirements. The RITE methodology and the evaluations performed also helped to refine them.

7.1.1.1. Functional requirements

- **Data overview:** users must be able to get global overviews of the data, useful to understand the data structure and find out what the data is about. Related with challenges 1, 2 and 3.
- **Data exploration:** users must be able to explore data through visual filters. They do not need to have knowledge of Semantic Web technologies. Related with challenge 4.
- **History of actions:** users must be able to see the history of actions and go back to previous pages. Related with challenge 5.
- **Data detail:** users must be able to get details of the data. It is necessary to identify suitable methods for presenting the data to all potential end users, hiding the complexity of RDF. Related with challenges 3 and 6.

7.1.1.2. Non-functional requirements

- **Support for scalability:** the IA components must be able to manage large amounts of complex and heterogeneous data. Related with challenge 8.
- **Non-domain specific:** the IA components must be compatible with multiple domains. Related with challenge 9.
- **Standards and browser compatibility:** the system should conform to established Semantic Web standards and be compatible with the main web browsers. Related with challenge 7.

7.1.2. Design

The first design choice is where to place facets and navigation menus within the window. There exist two basic approaches for both components: vertical or horizontal placement. Horizontal facet alignment limits the number of facets that can be displayed without scrolling. On the other hand, vertical alignment does not have such limitation because vertical scrolling is much more prevalent in web-based environments [NL06]. However, too much vertical scrolling introduces usability issues that should be considered. Moreover, if facets are placed on top, the user will have to scroll to see the results. Users should be able to see initial results immediately without requiring scroll [Hea06a].

7.1. ITERATION 1

Regarding navigation menus, user expect to find vertical navigation down the left side or horizontal navigation across the top or. Since the left side is reserved to facets, we decide to place navigation menus on top of the user interface. Putting both components in standard places should make our site easier to use.

Consequently, we have divided the user interface in three areas:

- An area located on the top containing navigation menus.
- A secondary facet area, located in the left side, where users perform selections of facets.
- A main area that shows the results of performing queries.

The main area presents a collection of RDF resources as a list. The previous version of *Rhizomk* already provides an HTML view on the data and also facilitates the navigation across the data graph. The main point of interaction is the list of facets, presented on the left side of the result list.

Figure 7.1 shows a paper prototype with the Information Architecture components proposed. This prototype does not correspond with the final implementation and was not used for user tests, but to place the components in the user interface.

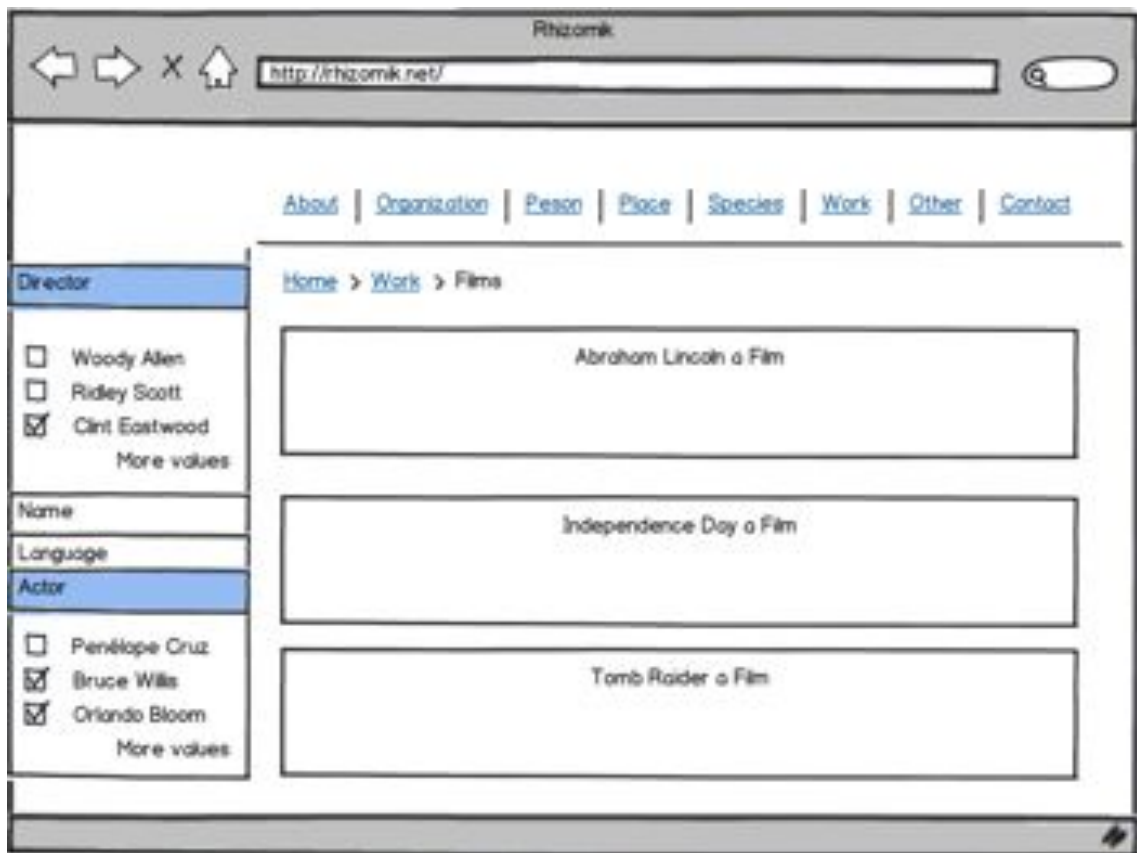


Figure 7.1: Paper prototype

7.1.3. Implementation

7.1.3.1. Navigation menus

Existing research suggests that navigation menus should contain a small number of options. Users can recall immediately only a small number of chunks, i.e. 7 ± 2 [Mil56]. This number should not be considered as something exact, but as orientative. Therefore, the objective is to generate a navigation menu that best represents the dataset and contains only a small number of classes.

We have implemented navigation menus with the following parameters:

- Two levels in the menu.
- Seven elements in each level of the menu.
- Displayed in alphabetical order.

7.1.3.2. Facets

Hearst [Hea08a, Hea06a] describes design guidelines for faceted search interfaces. He discusses some issues surrounding such as navigation, breadcrumb trails, keyword search and graphic design. We have followed these guidelines to implement our faceted browser, which is described next.

Cardinality data and previews

Many faceted browsers present an indication of the resource set size that would result from selecting one facet value. This normally takes the form of a numeric count. The counts on facet values provide guidance to the end user for further navigation. A user can see the operations that seem most promising and might be interested to choose the value that has the most resources.

Pre-populating every facet requires to perform several SPARQL queries and obtain all the object counts. These operations are expensive and counting can take several seconds to complete, making the browsing experience tiring. Previous research shows that users leave web sites when they take more than a number of seconds to load [NL06].

To overcome this situation, all facets are initially folded. When the user unfolds a facet, its values are loaded and presented to the user as a list of HTML checkboxes, thus allowing multiple selections. The values are ordered by the number of resources that correspond to that facet. Due to limitations of space, not all values can be presented. Therefore, only the top 5 values are displayed with the option to show other values by using a “more” button.

To address this issue, we also provide an autocomplete box to allow users to perform a text search for a particular value within a facet [BW06] without needing to look for it in the value list. Autocomplete or word completion is a feature in which, as the user is typing a term into an entry box, terms that are lexically related and highly relevant are shown beneath the entry form [W06].

7.1. ITERATION 1

Facets are rendered as HTML, as it shown in Figure 7.2. The interface shows a sidebar with the target facets selected using the metrics described in section 6.2. Each facet consists in a list with the five most used values and a text search box, which suggests possible matches. There is also the possibility to see the rest of values and choose from them.



Figure 7.2: Automatic facets for the <http://dbpedia.org/ontology/Film> class

SPARQL queries generation

Given a set of facet selections, the faceted browser constructs two types of queries required by the UI: those to obtain the values for each facet (and item counts), and those to acquire the resources of interest.

Our faceted browser is non-directional [WAs08], i.e. facets are treated independently of their order and selections in one facet affect other facets in the same way. Selecting a value from a facet modifies the content of the rest of facets with appropriate values, i.e. those that do not lead to dead ends. This is an important aspect of a facet browser. When constraining the dataset, all properties and values that would lead to an empty set of results need to be automatically removed from the interface, protecting the user against dead ends. Facets and values need to be changed on the fly. This can save users time.

Example 18 shows the SPARQL query to retrieve the 5 most common values for a class and property taking into account the filters applied so far. The filters applied so far by the users are converted into filters in the resulting SPARQL query. These filters are used both to compute the common values and to compute the instances of the class for which the faceted view is shown that satisfy all the selections applied so far.

```

1 SELECT ?o (COUNT(?o) AS ?n)
2 WHERE {
3   ?x a <%classURI%> ; <%propertyURI%> ?o .
4   %generateConstraints(facetsFilters)%
5 }
6 GROUP BY ?o
7 ORDER BY DESC(?n) LIMIT 5

```

Example 18: SPARQL query generated to obtain the 5 most common values for a class and property

The UI allows to perform simple operations on facets. Selecting multiple facets creates a conjunction, while selecting multiple values of a facet acts as disjunctive selections. The resulting query is limited to conjunctions of disjunctions, e.g. (Facet1=ValueX OR Facet1=ValueY) AND (Facet2=ValueZ). Example 19 shows the SPARQL query generated when the user has selected actor="Orlando Bloom" and director="Peter Jackson", i.e. those movies directed by "Peter Jackson" and starring "Orlando Bloom".

```

1 SELECT DISTINCT ?r1
2 WHERE {
3   ?r1 rdf:type <http://data.linkedmdb.org/resource/movie/film> .
4   ?r1 <http://data.linkedmdb.org/resource/movie/actor> ?r1var1 .
5   FILTER(str(?r1var1)="Orlando Bloom") .
6   ?r1 <http://data.linkedmdb.org/resource/movie/director> ?r1var2 .
7   FILTER(str(?r1var2)="Peter Jackson")
8 }

```

Example 19: SPARQL query generated to obtain results

More complex relationships - such as existential selection - would broaden the applicability of the faceted browser for creating even more complex queries. However, we believe that it would reduce the simplicity of using faceted browsing and could be counterproductive for the users.

7.1.3.3. Breadcrumbs

The three presented types of breadcrumbs can be implemented for the Semantic Web. In this iteration we have only implemented attribute breadcrumbs because they are essential for facets. The other types of breadcrumbs, location and path, will be considered in further iterations.

In faceted search interfaces, attribute breadcrumbs show selected facet values and allow users to remove the selected filters. Users must always know where they are in the faceted browser. It is important to show which facets they have chosen and they must be able to remove the selected filters. Attribute breadcrumbs allow to easily change the query to a previous state. The user can eliminate an entire facet or some selections by clicking on the "X" icon.

To implement breadcrumbs we store a collection of key-value pairs. For each selected property we store its URI as the key and a list of the selected values for that property. Breadcrumbs are displayed horizontally with each selected property in a different line. The order of the breadcrumbs reflects their order of selection.

7.1.4. Prototyping

The implementation in *Rhizomer* is composed of a front-end (UI) and back-end system. The UI is based primarily on HTML and Javascript together with the jQuery framework. The back-end is implemented in JAVA. The Jena API [CDD⁺04] is used to obtain the class hierarchy and generate the overview, which is stored in a RDF file using VOID vocabulary, as described in section 6.1. We also use this API to build the facets and rank them. What *Rhizomer* does is to perform SPARQL queries for each class in the dataset that retrieve all the properties and calculates the metrics described in section 6.2. Facets are pre-calculated and stored in a data structure. Their values are updated when the user starts browsing and selecting values for different facets.

Figure 7.3 shows the software prototype implemented and available at <http://rhizomik.net/linkedmdb01d>. This prototype was used for the evaluation with end users.



Figure 7.3: Software prototype

7.1.5. Evaluation

The developed Information Architecture components were tested with end-users in order to evaluate their functionality and usability. The goal of the test conducted so far was to do a preliminary evaluation of the Information Architecture components, determine whether or not they are understood and if they improve the user performance when looking for a specific piece of information.

7.1.5.1. Experimental Design

The tests were performed in the UsabiliLAB²² facilities at Universitat de Lleida. To register sessions we used Morae Recorder and Morae Observer to analyse user test data. We used a real test dataset called the Linked Movie Database (LinkedMDB)²³. LinkedMDB is generated from the Internet Movie Database (IMDb)²⁴, data is extracted from the IMDb site and represented as Linked Data. We chose the movies domain because it is well-known for most people and quite appealing.

The objectives of the evaluation were:

- To compare the navigation systems offered by *Rhizomer* with LinkedMDB and IMDb.com.
- To evaluate the Information Architecture components developed and the overall interaction of the *Rhizomer* platform.
- To obtain spontaneous feedback from the users and detect other usability problems that were not previously considered.

Six participants were selected, with a unique profile characterized by good knowledge of information technology, limited knowledge about Semantic Web technologies and interest in movies. They all fit in the lay-users profile. The number of participants was greater than 5, as recommended by Nielsen [NL94] for qualitative user tests. The participants were recruited and filled out a pre-test form. This information was necessary to determine if they belonged to the desired user profile. Then, they signed a confidentiality document, giving permission to be recorded.

The test had three phases:

- Free exploration of the prototype: the participant gets in touch with the system and explores it.
- Task realization: the participant receives a document with instructions for each task. He must perform the task and describe his interaction with the system. After performing each task, the participant must fill out a form about it.
- Interview and comments: after completing all the tasks, the participant is interviewed and he can give comments about the system and the tasks. He must also fill out a post-test form.

7.1.5.2. Tasks

We considered interesting to compare the evaluation results with those for IMDb and thus be able to test if we could obtain a similar user experience using Semantic Web data compared to the original web site. Consequently, we established one scenario with one task to be performed with IMDb and another one with one task for *Rhizomer*.

²²<http://griho.udl.cat/about/infrastructure.html>

²³LinkedMDB by O. Hassanzadeh and M. Consens, awarded 1st prize at the Linked Open Data Triplification Challenge 2008, <http://triplify.org/Challenge/2008>

²⁴<http://www.imdb.com/>

7.1. ITERATION 1

The test facilitator proposed users the two scenarios and tasks, but not necessarily in the same order to minimize the learning effect:

- **Task 1:** “Find three films where Woody Allen is director and actor at the same time” using IMDb.
- **Task 2:** “Find three films where Clint Eastwood is director and actor at the same time” using *Rhizomer*.

7.1.5.3. Usability metrics

Our evaluation was partially based on SWET-QUM, because the quality model was not already completed at the time of the evaluation. For the usability test we chose the following metrics:

- Effectiveness:
 - Task success.
- Efficiency:
 - Task time.
 - Number of help requests.
 - Task efficiency: percentage of task success per time.
- Context coverage:
 - Task flexibility.
 - Layout flexibility.

We did not consider user satisfaction yet because the focus at the current stage is on the user interaction when solving tasks. This metric will be considered in the next iterations as well as other metrics related to eye tracking.

7.1.5.4. Results and discussion

Effectiveness and Efficiency

Table 7.1 includes the measures of most of the metrics corresponding to the Effectiveness and Efficiency factors, except for UI Components Efficiency and Effectiveness, which were not considered in this evaluation. The table compares the task performed using IMDb and the task performed using *Rhizomer*.

| Task | Task success | Task efficiency | Task time (m) | Help requests |
|--------------------|--------------|-----------------|---------------|---------------|
| Task 1 IMDb | 100% | 32% | 3.37 | 5 |
| Task 2 Rhizomer | 100% | 54% | 2.41 | 6 |

Table 7.1: Evaluation results for iteration 1: effectiveness and efficiency

As it can be seen, both tasks were successfully completed by all users. However, only one participant was able to complete the first task without assistance while 100% of participants needed in at least one occasion the guidance of the facilitator to successfully complete the second task. In IMDb, users required help to find the list of films starring “Woody Allen” because the page gives priority to the films directed by him. In *Rhizomer*, 100% of the participants began the navigation from actors instead than from films. This was the reason why users required assistance but as soon as they realized they were able to start it from films, the task was easily solved. Considering that most of users required help for both tasks, 83% of them completed the second task in less time than the first. Only one user completed the first task in less time than the second. Overall, efficiency is relatively low for both tasks: 32% on average in the first task, and 54% in the second task.

Context coverage

For the Context Coverage quality factor, the Task Flexibility and Layout Flexibility properties were measured for *Rhizomer*. We have not consider these properties for IMDb because it is a traditional website not based on Semantic Web technologies. The measures for both metrics for *Rhizomer* are shown in Table 7.2.

| Metric | Value | Description |
|-----------------------|-------|---|
| Task Flexibility | 33.3% | It is only possible to complete the task starting from films |
| Layout Flexibility | 1.5 | Zoom film (0), expand actor facet (1), search box (2), select actor (3), expand director facet (1), select director (2) |
| Min Interaction Steps | 6 | |

Table 7.2: Evaluation results for iteration 1: context coverage

For Task Flexibility, we determined that, from the conceptual point of view, users should be able to perform the task following 3 main paths: starting from actor, starting from director or starting from film. However, *Rhizomer* only allows to perform the task starting from films. Therefore, the task flexibility is only 33.3%.

For Layout Flexibility, the task to perform with *Rhizomer* was analysed to determine the shortest interaction path to complete it and the depth at which each interaction step was performed. In this way, if the user uses a link in the entry page it is considered an interaction at depth zero. If the user expands a facet it is considered an interaction at depth one. Then, if the user uses the search box after expanding a facet it is considered an interaction at depth two.

7.1.5.5. Conclusions and proposals

The first tests with users show that *Rhizomer* facilitates publishing and browsing a dataset, like many other similar tools, but also allows users to realise what the value of the dataset is in the context of their particular needs. This is accomplished by the developed information architecture components.

From test results and their analysis, these proposals were elaborated to improve the IA components developed and the *Rhizomer* platform:

- The main issue detected is that the user interaction is currently too constrained by how the underlying data is structured. In this test, the task with *Rhizomer* was performed differently from how it was expected and this confused all users. They were looking for movies where actor and director were the same. Instead of initiating their interaction from the *Film* menu option, all users started from *Actor*. From there, as the underlying data just modelled actors per film but not the reverse, it was impossible to filter those films where the same person was the director. The easy way was to look for movies and to filter by director and actor using the corresponding facets, as the underlying data has these two properties associated to every film. The impression is that users tend to think firstly about persons and consider films a secondary entity. The idea here is to exploit the possibilities of the underlying conceptual model and derive implicit properties, for instance reverse properties, in order to provide users with alternative paths. In this particular case, there are reverse properties from actors to films.
- Navigation must be contextualised better. The interface should provide more mechanisms to inform the user where he is, where he can go and where he has been. For that, the proposal is to integrate breadcrumbs in natural language that summarise the navigation steps through navigation menus and facets.
- All items should be labelled so URIs or URI fragments are not shown to the user. For resources that have no label, this requires a tool that detects unlabelled items and creates a label for them automatically.
- A pagination mechanism is necessary to make it clear the total number of results and to allow users to browse them.
- Improve how facets are presented to the user, especially when there are a lot of facets and a lot of values for a concrete facet. For that, the proposal is to use values indexes or graphical representations for numeric values, e.g. histograms or sliders.
- Mark the external links, using some sort of image, text or colour, so in case the user leaves the application he is aware in advance.
- Hide some advanced features, like data edition, that are not useful for non-advanced users. Different user profiles should be defined and we will determine which options are displayed to each user profile.

7.2. Iteration 2

7.2.1. Requirements analysis

The evaluation performed in the first iteration showed mainly that users need more ways to interact with data and a better contextualization. A critical aspect of Semantic Web exploration tools is that they should be capable of making all the richness of the underlying data model available at the interaction level. They should not constrain the interaction, being flexible enough to offer users all the possible ways to conceive and interact with the data. However, this flexibility should not be at the expense of users' cognitive load. It should be possible to exploit data richness to provide a flexible interaction that satisfies different user mental models.

When analysing the evaluation results, it became evident that the fact that users started from actors was the reason why they required assistance. They arrived at a dead-end after filtering actors by name to just "Woody Allen" and there was no way to switch to his set of films and then filter it using the director facet. A short path to this problem might be to add for each resource, e.g. "Woody Allen (Actor)", a link for each facet to the set of resources that can be reached through it, e.g. a link to all the films where Woody Allen has acted. However, this just works for particular instances and the objective is to make it also work for sets of resources, e.g. all the films by a Spanish actor.

Another solution to this problem is to add to each class faceted-view some derived facets, i.e. facets from other classes that are directly connected to the current one through a property, as *tFacet* [BH11] does. This interface is similar to Windows Explorer with facets that expand and collapse, showing subfacets. This allows subfacets to be simultaneously viewable from root to the desired level. For instance, add the "directed by" facet to actors derived from the "director" facet of the films they have acted in.

However, this approach has at least three drawbacks. Firstly, if many facets are expanded, the number of facets for each class gets easily unmanageable and derived facets lose their context easily and become confusing. Therefore, facets can get very large and require large scrolling [Hea06a]. Secondly, subfacets might form cycles when items are related in different ways [CNF09]. Moreover, it can be difficult to distinguish between the "country" facet for author birthplace and a "country" facet derived from the country of the films the actor has participated in. Finally, with subfacets it is possible to apply filters from related resources, but it is still not possible to switch to that new set of related resources.

This motivates the development of a pivot operation that allows to switch between different types of resources. We also obtained the following formal requirements for this iteration:

- **Pivoting support:** it is necessary a mechanism that allows to switch between different related types in facet browsing, e.g. from actors to films. Related with challenge 4.
- **Better contextualization:** attribute breadcrumbs in facet browsing are not clear enough. It is necessary to improve the contextualization and show users where they are and what they are seeing in facets. Related with challenge 5.
- **Automatic generation of labels:** some resources do not have a label. In this cases, an automatic label must be generated using the resource's URI. Related with challenge 6.

7.2.2. Design and implementation

7.2.2.1. Pivoting in facets

Pivoting is defined as *“a way to restart a search from the results of a first search”* [SF09]. From the point of view of OLAP systems[CCS93], pivoting or rotation is described as *“an operation producing a change in the dimensional orientation of data”*. For instance, if data is initially aggregated by Product, Location and Date, by pivoting, the user can aggregate, for instance, by Location, Date and Product.

This operation is particularly important in the context of interactive semantic data exploration. Filtering just at the level of one class, using for instance facets, is not sufficient for many users. Users should be capable of building queries that mimic natural language relative sentences like *“photos of buildings in the town, where the WWW conference took place in 2004”*. In this case, the related classes are cities and conferences, the user must be able to filter both and relate them through a pivoting operation.

Usually, the type of resources to be browsed (e.g. film, actor, director...) remains fixed in a faceted browsing application. However, when pivoting is added to faceted navigation, it allows to switch the type of displayed entities based on relations to the current result set. For instance, a user who is filtering films using film facets, e.g. director is *“Woody Allen”*, then pivots on actors. As a result of this action, the user will see now all actors in the result list, who are related to any film in the previous filtered list. Then, the user can continue filtering but now using actor facets, e.g. country of birth is *“Spain”*.

Pivot steps can be repeated, e.g. pivot on countries of birth from actors and filter continent is *“Europe”*, after removing the previous country is *“Spain”* filter from actors. Each pivot step corresponds to a nested relative sentences, such as *“Show European countries, where an actor, which has acted in a Woody Allen film, has been born”*.

Pivoting implementation

The first step to implement pivot-enabled facets is to determine which ones should provide pivot. Properties with XML Schema data types or RDF Literal values, for a given class, result in facets that do not provide pivoting. On the other hand, properties that connect to other resources allow pivoting. To build facets that support pivoting, we first distinguish three types of properties:

- **Datatype properties:** properties whose values are RDF literals or data types from XML Schema. It is not possible to pivot on these properties but recognising them allows to display them with specialised facet types, e.g. a slider facet for numbers or a calendar one for dates. These specialised facets will be considered in further iterations.
- **Object properties:** properties whose values reference other resources. These properties were treated, prior to the introduction of pivoting, as facets with literal values, where the values were resources labels. It continues to be possible to filter a set of resources based on the labels of the referenced resources, e.g. filter films through the actor facet based on the actors' labels. However, pivoting makes also possible to switch to the set of actors and perform a more detailed filtering based on actors' facets.

- Inverse properties: in some cases, a dataset has a property between resource types modelled just in one way. For instance, each resource of type `Film` has the property `actor`, but the resources of type `Actor` do not have the inverse property to relate them with the films they have appeared in. When inverse properties are not explicit in a dataset, they are detected and facets are generated following the same approach than for explicit object properties. Consequently, it is possible to pivot through explicit object properties and also through implicit inverse object properties. This increases the flexibility of the exploration as more choices are available to the user dead-ends are avoided, like exploring actors and not being able to pivot to films because the property from actors to films is not explicitly stated in the dataset.

The properties that permit pivoting are those that reference to other subjects (object properties) or inverse properties. In these cases, it should be determined to which class the pivoting operation must refocus. The faceted view for that class will become the new view when pivoting is performed. This distinction is made by analysing the underlying dataset and ontologies. It results in an additional facet characteristic: its range. The procedure to determine a facet range is the following:

1. Check if, for the given class and property, there is an OWL restriction that defines the property range. This range is selected as the facet range. It can be either a class, a type from XML Schema or a RDF literal.
2. If no restriction is found in the previous step, it is checked if the property has a defined range, which becomes the property range.
3. If there is no property range, the dataset is analysed and the 5 most common values (Example 20) for the class and property are retrieved. They are checked to determine whether they are resources or not:
 - a) If all the 5 values are resources, then the dataset is queried to determine the most instantiated classes by the values of the property. At most five of them are retrieved using the query shown in Example 20. This list of common classes is then passed to the query in Example 21. This query retrieves the most specific superclass of all the input classes. The result is then considered the range of the facet and that class will become the new faceted view when the user pivots the facet.

```
1 SELECT ?type (COUNT(?type) AS ?n)
2 WHERE {
3   ?x a %classURI% ; %propertyURI% ?o .
4   ?o a ?type .
5 }
6 GROUP BY ?type
7 ORDER BY DESC(?n)
8 LIMIT 5
```

Example 20: SPARQL query that retrieves at most the 5 most common classes instantiated by the values of a facet

7.2. ITERATION 2

- b) If not all 5 values are resources, their data type, if present, is retrieved or computed by trying to parse their values as an integer, double or date. By default, the value is considered to be a string. Then, the range of the facet is set to the most specific super datatype in the XML Schema datatypes hierarchy²⁵. As no pivoting is enabled for this kind of facets, the range might be used to create specific facets such as sliders for numeric values or dates.

Rhizomer keeps track of all pivoting operations and records the initial class, the pivot property and the target class. Example 22 shows the SPARQL query generated when browsing films whose director is “Woody Allen”.

When pivoting to the new class, the restrictions applied to the previous ones are propagated to the pivoted class through the property used for pivoting and SPARQL variables. For example, when pivoting from films to actors:

- Initial type: `http://data.linkedmdb.org/resource/movie/film`
- Pivoting property: `http://data.linkedmdb.org/resource/movie/actor`
- Pivoting type: `http://data.linkedmdb.org/resource/movie/actor`

The constraints capturing the pivoting switch are introduced in the generated query, as shown in Example 23 in lines 3-4. A new variable `r2` is introduced together with its type, i.e. the range of the originating facet. The link from the previous variable `r1` to the new one is established using the pivoted property, i.e. `movie:actor`. Finally, the selected variable is the new variable as the focus has changed from films to actors.

Figure 7.4 illustrates the different sets of resources that are selected from Films by filtering those directed by “Woody Allen” and then the set of Actors selected after pivoting from the previous set of Films through the Film facet corresponding to the actor property.

```
1 SELECT DISTINCT ?common
2 WHERE {
3   ?common rdfs:subClassOf %listOf5CommonClasses%
4   OPTIONAL {
5     ?intermediate ^rdfs:subClassOf %listOf5CommonClasses%
6     ?intermediate rdfs:subClassOf ?common.
7     FILTER (?intermediate!=?common && !isBlank(?intermediate)) }
8   FILTER (!BOUND(?intermediate) && !isBlank(?common))
9 }
```

Example 21: SPARQL query that computes the most specific common superclass

²⁵<http://www.w3.org/TR/xmlschema11-2/#built-in-datatypes>

```

1 SELECT DISTINCT ?r1 WHERE {
2 ?r1 a <http://data.linkedmdb.org/resource/movie/film> .
3 ?r1 <http://data.linkedmdb.org/resource/movie/director>
4 <http://data.linkedmdb.org/resource/director/8501> }

```

Example 22: Generated SPARQL query before pivoting

```

1 SELECT DISTINCT ?r2
2 WHERE {
3 ?r2 a movie:actor .
4 ?r1 movie:actor ?r2 .
5 ?r1 a movie:film .
6 ?r1 movie:director <http://data.linkedmdb.org/resource/director/8501>
7 }

```

Example 23: Generated SPARQL query after pivoting

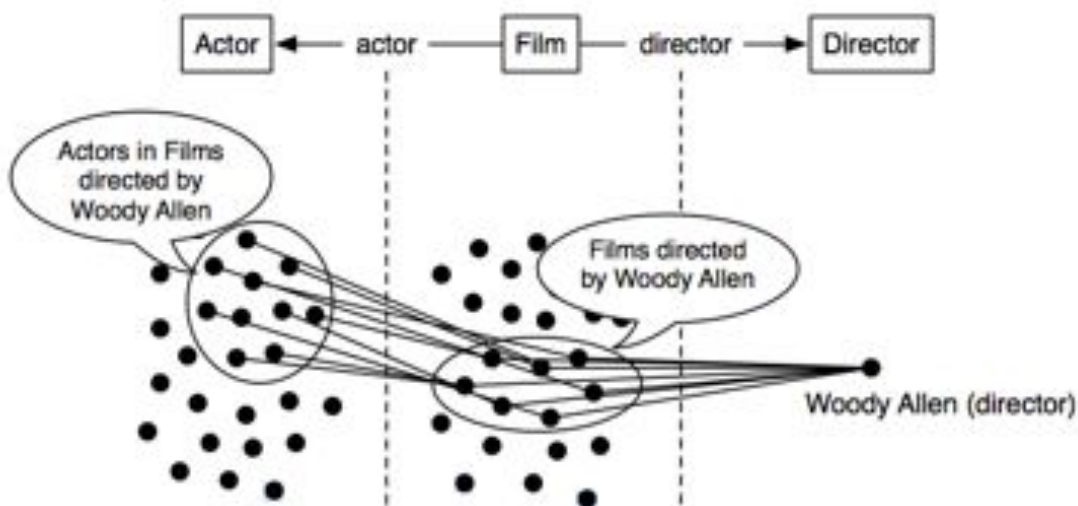


Figure 7.4: Set-based browsing through pivoting

7.2.2.2. Literal breadcrumbs

The implementation of pivoting is powerful and allows users to make more complex queries between different types of resources. However, breadcrumbs become also more complex and difficult for users to understand. The pivot operation also encouraged us to consider some sort of breadcrumbs that help contextualise user interaction. Previous tests with users showed that they got lost easily after moving around the underlying graph models. Breadcrumbs should show the path that the user has followed to arrive to the set of results that is currently displayed. Users should also be capable of using the breadcrumbs to undo previous filtering and pivoting steps.

The resemblance between pivoting and natural language can be used to generate more usable breadcrumbs that help users contextualise their exploration. Indeed, the query above can be rephrased as “Showing actors, which have acted in films directed by Woody Allen”. The idea of pivot is reflected by the fact that the set of “actors” in the main sentence also appears in the relative sentence as the relative pronoun “which”. The relative pronoun points to the facet to browse for a pivot, in this case “acted in”.

7.2. ITERATION 2

Currently, breadcrumbs have been implemented as a natural language representation of the SPARQL query generated as a result of the user interaction so far. In this way, users know why they are getting the list of results that they are seeing and see the pivoting steps performed so far.

7.2.2.3. Labels

Applications for linked data consuming are intended to be widely used by different kinds of users. Therefore, hiding technical details such as URIs when displaying data to users becomes crucial. Entities in the Semantic Web need to have labels in order to be showed to humans in a meaningful way. Labels are used for displaying the entities when exploring the data instead of displaying the URIs. They can also be used to support keyword-based or natural-language-based search. The property `rdfs:label` is usually used to provide a human-readable version of the resource's name besides its URI[?].

However, it is necessary to generate a label for those resources without these property. In these cases, the last part of the URI identifying that resource can be used to generate a label [EVS11]:

1. If the URI contains a local name, the last part of the URI is used. For example, for the URI `http://dbpedia.org/resource/Berlin` the last part of the path is used, i.e. Berlin.
2. If the URI contains a fragment identifier, the last part of the URI is used. For example, for the URI `http://www.example.com/about#Bob` the last part of the path is used, i.e. Bob.
3. When the URI contains the underscore character, it should be replaced with a space. For example, the URI `http://dbpedia.org/resource/Star_Wars` should generate the label "Star Wars".
4. When the URI contains a word with capital letters, in most cases it can be separated in two words. For example, a URI containing `hasPhotoCollection` should generate the label "Has Photo Collection".

7.2.3. Prototyping

Figure 7.5 shows our prototype with the user interface components used to enable the pivoting operation. Once the facets that should provide pivoting are determined, this option is offered to users as part of the facet using an arrow shaped link. Those facets that allow pivoting are also listed in a "Navigate to" box, showing related items and possible destinations. It also shows an example of breadcrumbs as natural language rendering of the executed query.

7.2.4. Evaluation

The aim of the evaluation was mainly to validate that the introduction of pivoting solves the problems highlighted in the previous evaluation. Moreover, we were also interested in comparing *Rhizomer* with other related tools that provide pivoting.



Figure 7.5: Pivoting enhancements

7.2.4.1. Experimental Design

The tests were performed in the UsabiliLAB²⁶ facilities at Universitat de Lleida. The experimental setting used two computers. One of them was for the user and it is equipped with an Eye-Tracker and Morae Recorder, which registered user interaction, where the user is looking at, screen video, clicks, mouse position, user voice and user video through a webcam. The other is equipped with Morae Observer and Morae Manager, which were used by the evaluation team to observe, annotate and analyse the interaction session.

As reviewed in the Related Work section, the only active tools capable of providing a faceted view and pivoting on semantic data are *SParallax* [HK09], *Virtuoso Faceted Browser* [Er], *Visor* [PSHS11] and *gFacet* [HZL08]. We have not considered *tFacet* because it only provides subfacets but does not allow to refocus on a related set of resources. Among these tools, we have selected *SParallax* and *Virtuoso Faceted Browser* to make a comparison. Both tools provide a user interface with HTML and components similar to those that we propose. We have not included *Visor* and *gFacet* because their interfaces are based on graphs and we believe that they are not suitable for lay-users. All tools were deployed on top of the LinkedMDB dataset.

The objectives of the evaluation were:

- To evaluate pivoting and test whether or not it solves the problems detected in the first evaluation.
- To compare the navigation systems offered by *Rhizomer* with *SParallax* and *Virtuoso Faceted Browser*.
- To detect other usability problems and obtain spontaneous feedback from the users.

Overall, 13 users were involved, all of them in the lay-user profile. 2 groups were defined: 7 users for the *Rhizomer* test and 6 users to test *SParallax* and *Virtuoso Faceted Browser*. The groups involved always more than 5 users as recommended by Nielsen [NL94] for qualitative user tests. None of the users received any a priori training about the evaluated tools.

²⁶<http://griho.udl.cat/about/infrastructure.html>

7.2. ITERATION 2

The evaluation process was conducted based on a mix of evaluations and questionnaires. The evaluations with users were based on tasks to be completed using the evaluated Semantic Web exploration tools. Then, the interaction was analysed and the selected metrics among the proposed set were used to measure the quality factors of each evaluation. The evaluations with users were complemented with questionnaires that measured the satisfaction factor and collected information about their perception or the process of use, the hedonic and subjective quality.

7.2.4.2. Tasks

Since the main objective of the test was to validate if there was improvement with pivoting, we considered important to keep one task from the first evaluation. Therefore, task 2, was identical to one used in the previous evaluation round. It was used to test if pivoting had improved the efficiency and efficacy. The complete set of tasks was:

- **Task 1:** find 5 films with “Orlando Bloom” as actor.
- **Task 2:** find 5 films with “Clint Eastwood” both as director and actor.
- **Task 3:** find who has directed more films in countries located in “Oceania”.

The tasks were given with increasing difficulty. The objective of the first task was to introduce the user to the system. Therefore, we did not measure the metrics for it. It was a simple task that could be performed without pivoting. Task 2 could be also be performed without pivoting, but this operation provided users more paths to complete this task. Finally, pivoting was necessary to perform task 3.

7.2.4.3. Usability metrics

We used SWET-QUM as the framework for our evaluation. For the usability test we chose the following metrics:

- Effectiveness:
 - Task success.
 - UI Component effectiveness.
- Efficiency:
 - Task time.
 - Number of help requests.
 - Task efficiency: percentage of task success per time.
 - UI Component Efficiency.
- Context coverage:
 - Task flexibility.
 - Layout flexibility.
- Satisfaction.

| | | Task 2 with pivoting | Task 2 pre-pivoting | Improvement | Task 3 with pivoting |
|------------------|----------------------|-------------------------|------------------------|-------------|-------------------------|
| Task time (m) | Minimum | 0.89 | 1.05 | 15% | 1.99 |
| | Maximum | 2.23 | 5.23 | 57% | 4.50 |
| | Mean | 1.69 | 2.41 | 30% | 3.43 |
| | Standard Dev. | 0.57 | 1.49 | 62% | 0.96 |
| Task success | Without assistance | 100% | 0% | 100% | 0% |
| | Including assistance | 100% | 100% | 0% | 100% |

Table 7.3: Comparison with previous evaluation

7.2.4.4. Results and discussion

After completing the evaluation process, the metrics were analysed in order to compare the three tools. The first part of the analysis consisted in comparing the effectiveness and efficiency of *Rhizomer* with its previous version from iteration 1. Then, in the second part of the analysis we compare the three tools evaluated at three levels. The first one corresponds to the effectiveness and efficiency quality factor, the second to context coverage and the third to satisfaction.

Effectiveness and Efficiency

Table 7.3 shows the results comparing Task 2 with our previous version of *Rhizomer* in iteration 1. The first finding is that the introduction of pivoting corresponded to a great increase of efficiency, with a 30% reduction in the mean time necessary to complete Task 2. However, the most promising outcome is that the biggest improvement has been in the reduction of the maximum time on tasks, with 57% improvement. From the point of view of effectiveness, it is important to highlight that all users completed Task 2 without facilitator help, while in the previous iteration, for the same task, all users required facilitator assistance.

This is related with the fact that, thanks to pivoting, all users were able to find their path to solve the task without requiring assistance. Contrary to pre-pivoting tests, where most users got lost when trying to complete the tasks starting from actor or director instead of from film, with pivoting all users were able to complete the task independently of their starting point without assistance. Consequently, the maximum time was reduced significantly.

Table 7.4 compares *Rhizomer*, *SParallax* and *Virtuoso Faceted Browser*. It includes the minimum, maximum, mean and standard deviation for the measures of most of the metrics corresponding to the Effectiveness and Efficiency factors, except for UI Components Efficiency and Effectiveness that are discussed later.

7.2. ITERATION 2

| Tool | Metric | Task 2 | | | | Task 3 | | | |
|-----------------|---------|--------------|---------------|-----------------|---------------------------|--------------|---------------|-----------------|---------------------------|
| | | Task Success | Task Time (m) | Task efficiency | Facilitator help requests | Task Success | Task Time (m) | Task efficiency | Facilitator help requests |
| Rhizomer | Min. | 100% | 0.89 | 45% | 0.00 | 100% | 1.99 | 22% | 1.00 |
| | Max. | 100% | 2.23 | 112% | 0.00 | 100% | 4.50 | 50% | 1.00 |
| | Mean | 100% | 1.69 | 68% | 0.00 | 100% | 3.43 | 32% | 1.00 |
| | St.Dev. | 0% | 0.57 | 30% | 0.00 | 0% | 0.96 | 10% | 0.00 |
| Virtuoso Facets | Min. | 0% | 1.61 | 0% | 0.00 | 0% | 2.83 | 0% | 0.00 |
| | Max. | 100% | 19.95 | 31% | 4.00 | 100% | 23.33 | 35% | 5.00 |
| | Mean | 42% | 10.65 | 7% | 2.33 | 58% | 12.44 | 10% | 2.50 |
| | St.Dev. | 49% | 6.44 | 12% | 1.63 | 38% | 8.99 | 13% | 2.26 |
| SParallax | Min. | 50% | 1.58 | 9% | 0.00 | 0% | 8.60 | 0% | 1.00 |
| | Max. | 100% | 6.26 | 32% | 2.00 | 100% | 12.02 | 12% | 3.00 |
| | Mean | 75% | 4.36 | 19% | 1.00 | 33% | 9.46 | 4% | 2.17 |
| | St.Dev. | 27% | 1.66 | 9% | 0.89 | 41% | 1.89 | 5% | 0.98 |

Table 7.4: Evaluation results for iteration 2: effectiveness and efficiency

The best results for each metric are highlighted in grey and bold. As it can be seen, *Rhizomer* shows the best or equal values for all metrics except in Facilitator Help Requests for Task 3, in which the best value is for *Virtuoso Faceted Browser*. For Task Success, Task Efficiency and Help Requests, *Rhizomer's* values seem significantly better, or at least comparable, than those for the other tools.

It has been possible to perform a statistical analysis comparing *Rhizomer* with *SParallax* and *Virtuoso Faceted Browser* for the Task Time metric. The statistical tests, based on independent t-tests, show that with a 95% confidence interval, *Rhizomer* Task Time is smaller for tasks 2 and 3 than for both *SParallax* and *Virtuoso Faceted Browser*. First of all, Shapiro-Wilk tests were used to check if the values for the Task Time metric were normally distributed for all tools and tasks. These tests start from the hypothesis that the data comes from a population with a normal distribution. Using Shapiro-Wilk, it was not possible to refute this hypothesis because the p-values generated by the tests are in all cases greater than 0.05, i.e. $p\text{-value} > 0.05$:

- *Rhizomer* Task 2: $p\text{-value} = 0.0813$, Task 3: $p\text{-value} = 0.5874$.
- *Virtuoso Faceted Browser* Task 2: $p\text{-value} = 0.6377$, Task 3: $p\text{-value} = 0.5365$.
- *SParallax* Task 2: $p\text{-value} = 0.6325$, Task 3: $p\text{-value} = 0.3988$.

Consequently, as they seem normally distributed, it was appropriate to apply one-sided t-tests to compare them and check to what level we can say that *Rhizomer* is more efficient than *Virtuoso FCT* and *SParallax*, i.e. the Task Time for *Rhizomer* is smaller than for the other two tools. For Task 2, comparing *Rhizomer* versus *Virtuoso Faceted Browser* using the Welch Two Sample t-test results in $p\text{-value} = 0.1008$, greater than 0.05. Therefore, it cannot be concluded, with a 95% confidence, that the time to complete the task with *Rhizomer* is significantly smaller than for *Virtuoso FCT*. The alternative test method, Wilcoxon, is not conclusive either.

This is mainly because there are just 3 valid Task Time measures for *Virtuoso* and the rest of the users did not complete the task. In any case, the success for *Rhizomer* is 100% while for *Virtuoso* it is 42% and, as it can be observed in the left box plot in Figure 7.6, *Rhizomer* seems more efficient than *Virtuoso*.

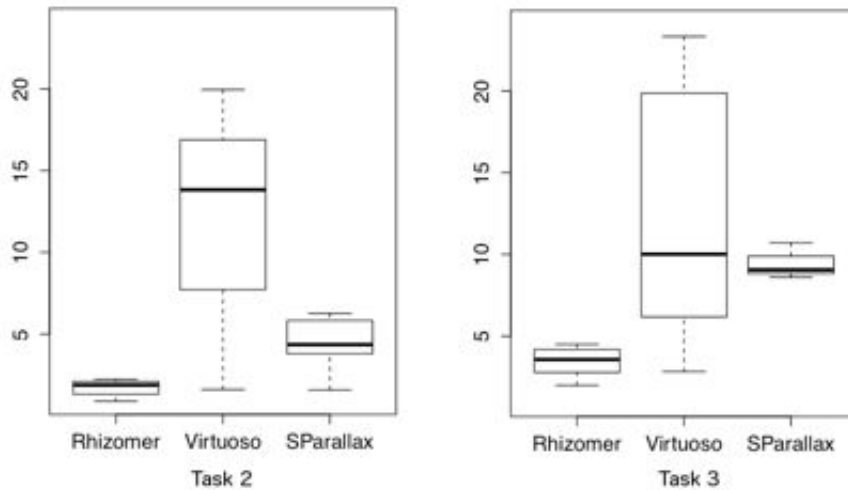


Figure 7.6: Statistical analysis, iteration 2

When comparing *Rhizomer* and *SParallax* for Task 2, the t-test p-values is 0.0048 so, with a 95% confidence interval, it can be concluded that the time to complete the task with *Rhizomer* is significantly smaller than for *SParallax*. Consequently, *Rhizomer* can be considered more efficient for this task than *SParallax*.

For Task 3, the t-test for *Rhizomer* versus *Virtuoso* results in p-value = 0.04204, which is smaller than 0.05 so it can be concluded that *Rhizomer* is more efficient than *Virtuoso* for this task. Finally, when comparing *Rhizomer* versus *SParallax*, the t-test p-value is 0.0012. As it is smaller than 0.05, it can be also concluded, with a 95% confidence, that *Rhizomer* is also more efficient than *SParallax*. The box plots for the Task Times for all tools and both tasks are shown in Figure 7.6.

UI Component Effectiveness and Efficiency

Table 7.5 presents the values for the remaining effectiveness and efficiency metrics. For all tools, the Data Exploration UI Effectiveness is 100% because all relevant UI components for data exploration did receive some attention by users during the evaluations. However, it is important to notice that when considering UI Components Efficiency, there are components that received really little attention (highlighted with light grey for each tool) while others received a lot (highlighted with dark grey).

To illustrate the data from the eye-tracker used to compute the UI Component Efficiency metric, Figures 7.7, 7.8 and 7.9 show the heat maps of *Rhizomer*, *Virtuoso Faceted Browser* and *SParallax*. They represent how user attention was distributed across the user interface. The figure also shows the location of the main relevant UI components for the proposed tasks.

7.2. ITERATION 2

The difference between the more attractive and less attractive components is especially significant in the case of *Rhizomer*, where facets received 70% of the attention while the “pivot button” and “navigate to” box only 13% together. This metric highlights a potentially problematic issue because these components were crucial for completing the tasks, especially those that required users to perform pivot operations to complete them.

For *Virtuoso Faceted Browser* there are also significant differences between the most attractive component, the Resource List, and the least one, the Resource Label. *SParallax* had the more balanced user interface from the UI Component Efficiency metric perspective. Moreover, the least attractive component is the Search Box, which is something natural as it was just used at the beginning of the tasks.

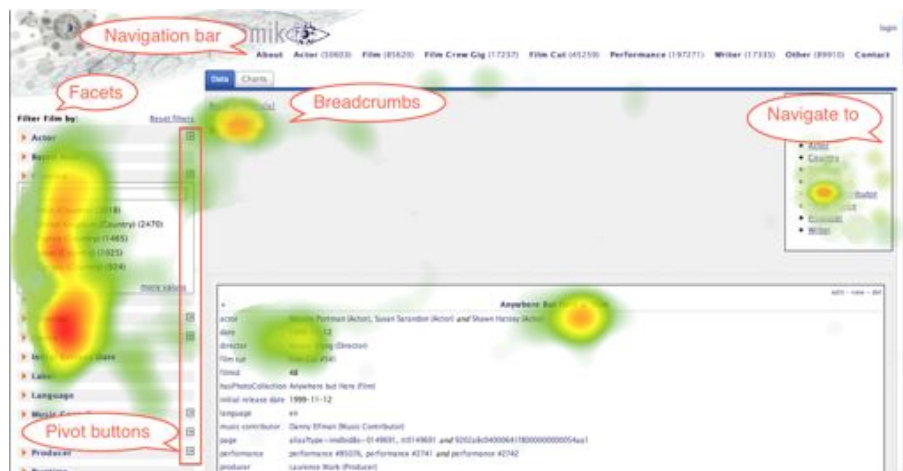


Figure 7.7: Rhizomer heat map, Iteration 2

| Tool | Relevant UI Components | UI Component Efficiency (%) | Data Exploration UI Effectiveness |
|------------------------|------------------------|-----------------------------|-----------------------------------|
| Rhizomer | Global Navigation Bar | 6% | 100% |
| | Facets | 70% | |
| | Facets "pivot button" | 4% | |
| | "Navigate to" Box | 9% | |
| | Breadcrumbs | 11% | |
| Virtuoso Facets | Breadcrumbs | 19% | 100% |
| | Navigation | 12.3% | |
| | Resource list | 56% | |
| | Resource properties | 12.4% | |
| SParallax | Breadcrumbs | 8% | 100% |
| | Connections | 21% | |
| | Facets | 28% | |
| | Search | 3% | |
| | Resource list | 24% | |
| | Resource properties | 16% | |

Table 7.5: Evaluation results for iteration 2: UI component effectiveness and efficiency



Figure 7.8: Virtuoso Faceted Browser heat map



Figure 7.9: SParallax heat map

Context coverage

For the Context Coverage quality factor, the Task Flexibility and Layout Flexibility properties were measured. The measures for both metrics are shown in Table 7.6. The best values for these metrics are marked in grey and bold in the table. For Task 2 Flexibility, it was determined that, from the conceptual point of view, users should be able to perform the task following 3 main paths: starting from actor, starting from director or starting from film. For Task 3 Flexibility, users should be able to complete the task following 4 main paths: starting from director, film, country or continent. For Layout Flexibility, both tasks were analysed to determine the shortest interaction path to complete them and the depth at which each interaction step was performed.

7.2. ITERATION 2

Compared to our previous evaluation, the Task Flexibility for Task 2 has increased. In *Rhizomer's* first evaluation, this task could only be completed starting from films, while now users can also start from actors and directors. However, for Task Flexibility for Task 3 is has the worst value compared to other tools. The only metric for which *Rhizomer* shows the best value is Layout Flexibility for Task 2. Finally, the amount of Interaction Steps required to complete both tasks should be improved because in both cases it is worse than for *SParallax*, the best tool in this respect.

A worse value in these metrics does not imply a worst user experience, at least from the point of view of the effectiveness and efficiency factors, as it has been already shown in the previous section that *Rhizomer* is more efficient and effective. However, if *Rhizomer's* user interface is improved from the point of view of the Context Coverage metrics, this could also produce improvements in the efficiency of the tool. This will be checked with additional evaluations based on SWET-QUM after changes are made.

Satisfaction

A questionnaire was presented to users after completing each task. They had to rate from 1 to 5 these particular questions:

- **TA1** The task was... (1) very hard - (5) very easy.
- **TA2** I think that I have done the task... (1) not correctly at all - (5) absolutely correct.
- **TA3** The interface structure... (1) did not help me at all - (5) did help me very much.
- **TA4** The time to complete task has been... (1) long - (5) short.

| Tool | Metric | Task 2 | | Task 3 | |
|--------------------------|--------------------|--------|--|--------|--|
| Rhizomer | Task Flexibility | 100% | It is possible to go through actor, director or film | 50% | It is possible to go through director or film, but not country or continent because they are not in the navigation menu. |
| | Layout Flexibility | 1.50 | Zoom film (0), expand facet (1), search box (2), select person (3), expand facet (1), select person (2) | 1.83 | Zoom film (0), pivot country (1), expand continent (2), select Oceania (3), pivot film (2), expand director (3) |
| | Interaction Steps | 6 | | 6 | |
| Virtuoso Faceted Browser | Task Flexibility | 100% | It is possible to go through actor, director or film | 100% | It is possible to go through director, film, country or continent. |
| | Layout Flexibility | 3.50 | Search "Woody Allen" (0), referring attributes (1), select actor (2), attributes (3), select director (4), distinct values (5), select "Woody Allen (Director)" (6), select "Entity 2" for films (7) | 2.50 | Search "OC" (0), referring attributes (1), select country (2), attributes (3), select director (4), distinct values (5) |
| | Interaction Steps | 8 | | 6 | |
| SParallax | Task Flexibility | 100% | It is possible to go through actor, director or film | 75% | It is possible to go through director, film or country. Not through continent because there are no facets for literals. |
| | Layout Flexibility | 2.00 | Search "Woody Allen" (0), select actor (1), more connections (2), select "actor of" (3), filter director (4) | 1.60 | Search "Country" (0), select country (1), filter "OC" (2), more connections (2), select "country of" (3) |
| | Interaction Steps | 5 | | 5 | |

Table 7.6: Evaluation results for the iteration 2: context coverage

- **TA5** To achieve the task I have had to be... (1) very focused - (5) not focused at all.
- **TA6** The task was... (1) badly defined, I did not understand the objective - (5) well defined, I understood the objective.

Additionally, the following questionnaire was presented to users after they had completed all tasks with a particular tool:

- **TE1** It is easy to use the tool - (1) disagree ... (5) agree.
- **TE2** The system is intuitive - (1) disagree ... (5) agree.
- **TE3** I had fun using it - (1) disagree ... (5) agree.
- **TE4** The options are easily identifiable - (1) disagree ... (5) agree.

The results for the post-task satisfaction questionnaire are shown in Figure 7.10, which compares the post-task satisfaction for both tasks and the three tools. As it can be observed, the satisfaction measures for each post-task question are clearly better (the higher the better) for *Rhizomer* in the case of Task 2 in comparison with both *Virtuoso* and *SParallax*. In fact, the results for *Virtuoso* are really low. In the case of Task 3, *Rhizomer* continues being perceived better when compared with *SParallax* but quite similar to *Virtuoso*, which improves from Task 2.

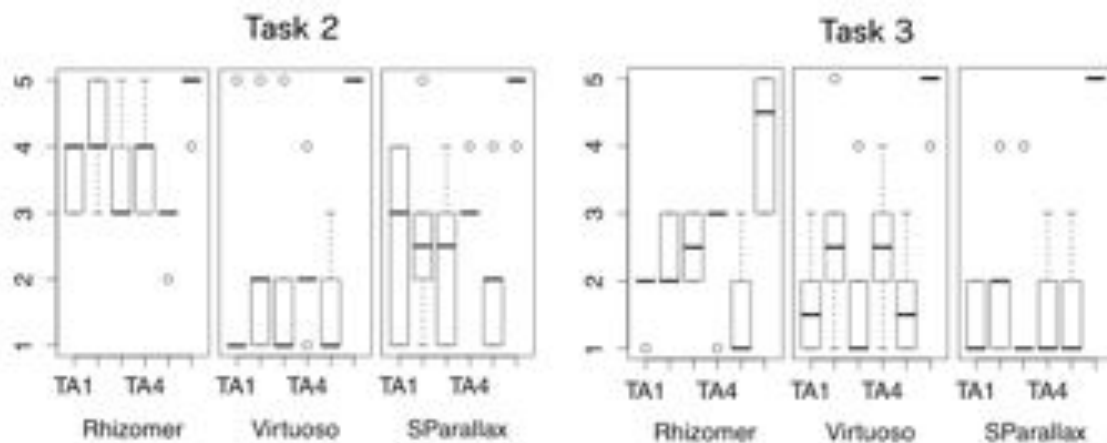


Figure 7.10: Post task satisfaction measures for iteration 2

Considering the feedback received during the evaluation, the reason for the reduced satisfaction in the case of *Rhizomer* and *SParallax* seems to be due to the fact that Task 3 was more complex than Task 2. For *Virtuoso*, there seems to be a learnability effect that explains the increased satisfaction. For Task 2, always performed before, users were quite unsatisfied with *Virtuoso* because it is hard to learn, as the biggest Task Times also indicate. However, some of the users were able to learn how *Virtuoso* worked during tasks 1 and 2 and then their satisfaction increased in Task 3, as they were able to successfully put into practice what they learnt. In any case, despite being more satisfied, *Virtuoso* is clearly less efficient than the other two tools and the satisfaction for Task 2 is quite similar for all tools.

7.2. ITERATION 2

After completing all tasks and post-task questionnaires, users also filled a post-test questionnaire that tried to capture their overall satisfaction with the user experience for each tool. The results for the post-test satisfaction questionnaire are shown in Figure 7.11. As it can be observed, the satisfaction measures for *Rhizomer* are the best ones and for *Virtuoso*, despite the improvements perceived in Task 3, are the worst ones.

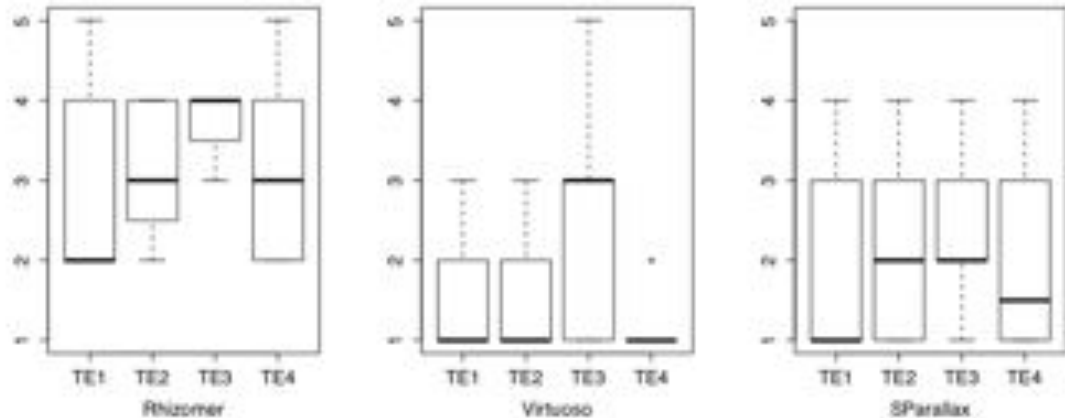


Figure 7.11: Post test satisfaction measures for iteration 2

7.2.4.5. Conclusions and proposals

Our second evaluation shows that there has been an improvement compared with the first version of *Rhizomer*, especially in efficiency and task flexibility. Thanks to pivoting, the tool offers more flexibility and it is possible to perform complex queries. Contrary to the tests prior to the introduction of pivoting, all users were able to complete task 2 without assistance.

However, because the pivot operation was not fully understandable, there is still room for improvement. The following issues were detected and some proposals to solve them are presented to be considered in further iterations of the development:

- Pivoting is not a common feature and supposes a conceptual shift that should be mitigated. For instance, some users understood pivoting as restarting the exploration for a new class of resources. With pivoting, users must refocus on a new type, make selections on its facets, and then refocus back to the original items. Pivoting causes every component on the screen to update and some users feel disoriented. This operation should be better explained to users.
- The interface components providing pivoting are not so evident for users. It was difficult for users to identify the pivoting button. Moreover, the box labelled “Navigate to”, that also contained the list of facets that provided pivoting, was far from the facets and hard to identify. These components should be more prominent on the user interface.

- Users also experienced many contextualisation problems, not being completely obvious for them what was presented to them at the screen. The breadcrumbs helped solving this once the users realised they were available. However, it took some time for most of the users to understand this. Moreover, breadcrumbs only show the executed query but not the pivoting steps. Breadcrumbs should be highlighted in the user interface and incorporate the pivoting path.

7.3. Iteration 3

7.3.1. Requirements analysis

Our evaluations so far have been focused on exploratory search tasks that users perform mainly using facets and breadcrumbs. In these tasks, users only use navigation menus to select one of the main classes to start navigation from it. Navigation menus are quite effective because lay-users are comfortable with them, most website feature them and they are used to interacting with them. However, they just provide an overview of the most frequent classes, those more instantiated. Other classes with less instances are not available from this component. This was the case in task 3 of our last evaluation, where the class *Continent* was not available from navigation menus.

In order to gain a more detailed overview, web sites usually apply the Directory Navigation pattern through different sorts of sitemaps and other components. The Directory Navigation pattern proposes to organize the items or pages into several groups. This pattern should be implemented when there are a large number of items, so users can select and focus on an item out of a large set.

Therefore, the following formal requirements have been obtained for this iteration:

- **Improve data overview:** users should be able to get a better overview of the data, understand the overall data structure and be able to easily find particular classes of interest. Related with challenges 1, 2 and 3.

7.3.2. Design and implementation

To support this overview we propose four components that are created from the generated hierarchy structure, described in section 6.1. Next, we describe the overview components implemented:

7.3.2.1. Navigation menus

Our implementation of the navigation menus is the same as it was in Iteration 1 (section 7.1).

7.3.2.2. HTML site maps

Since the components we propose should be able to deal with large datasets, instead of containing links to all the pages, the site map lists the main pages of the site. In this case, the site map shows the different classes from the hierarchy but not their resources.

CHAPTER 7. ITERATIVE USER INTERFACE DEVELOPMENT

We have implemented two different versions of the sitemap and users can switch between them. The first one is related with the structure of the generated navigation menus. It complements the main site navigation and users can find there options that were not directly available from navigation bars. It can be seen as a summarized version of the complete hierarchy structure. The second one reflects the original hierarchy of the dataset. It is displayed as a tree with multiple levels and the users can expand it. Figure 7.12 shows the summarized site map while Figure 7.13 shows the full site map



Figure 7.12: Summarized site map



Figure 7.13: Full site map

7.3. ITERATION 3

7.3.2.3. Site index

Our site index also shows all the classes in the dataset but organized alphabetically. The site index shows all letters belonging to the ISO basic Latin alphabet²⁷ “A-B-C-D-E-F-G-H-I-J-K-L-M-N-O-P-Q-R-S-T-U-V-W-X-Y-Z”. Numbers are grouped into the category “0-9” and other letters or symbols are grouped into a special category “*”.

While site maps can give users contextualization and users can understand the dataset structure, site indexes provide no context. Non-related categories appear in the site index without giving users any additional information. Therefore, we have implemented the site index so that it also provides context information of each class. When the user moves the pointer over an element, an overlay appears showing its parent and its subclasses.



Figure 7.14: Site index implementation

²⁷http://en.wikipedia.org/wiki/ISO_basic_Latin_alphabet:

7.3.2.4. Treemap

We have implemented the treemap visualization following recommended design issues [TJ92] and using the *JavaScript Infovis Toolkit*²⁸. The treemap shows two levels in the hierarchy and users can zoom in. The size of rectangles is proportional to the number of instances of each class. The nesting offset was 2px to put emphasis on the difference between levels. We used squarified treemaps as the tiling algorithm [BHvW00]. This algorithm produces rectangles with low aspect ratio, which makes it easier to compare their sizes. Different colors are used for each class and its subclasses. Label size is also proportional to the number of instances of each node. Figure 7.15 shows the treemap for the DBpedia dataset and Figure 7.16 shows the result after selecting the class “Work”.

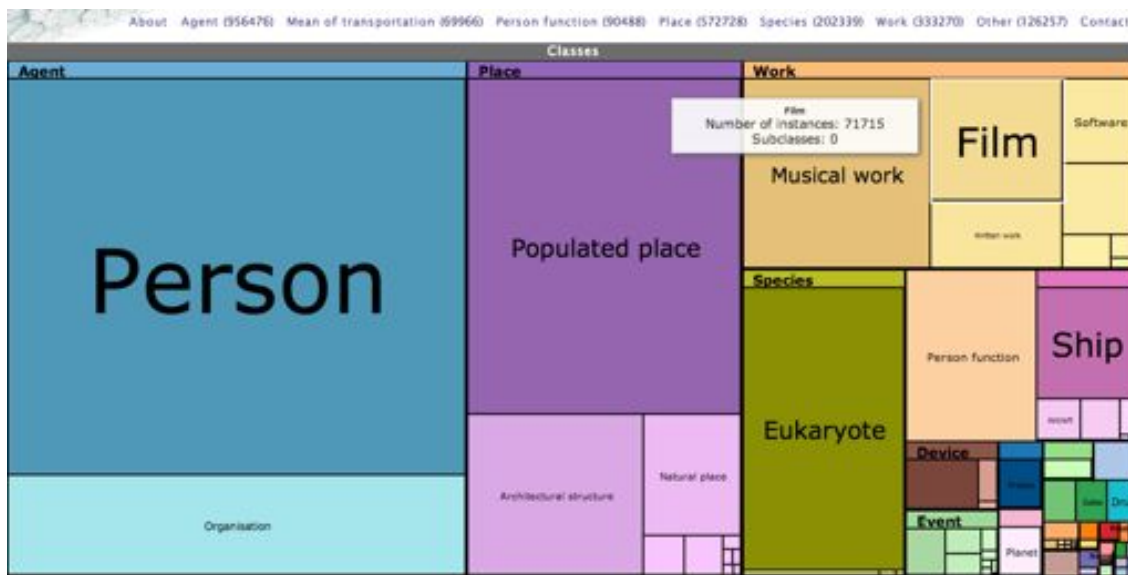


Figure 7.15: Treemap of the DBpedia dataset

7.3.3. Evaluation

The goal of the evaluation was to determine if solving tasks with the 4 different overview components differs with respect to task completion times, response accuracy and user satisfaction. The objective is to test whether or not users can quickly get an overview with different components. Comparisons between the different components can provide valuable information about the effectiveness and usability of these systems depending on the use case. We do not aim to face the systems against each other but to provide a basis for design recommendations and guidelines.

²⁸<http://thejit.org/>

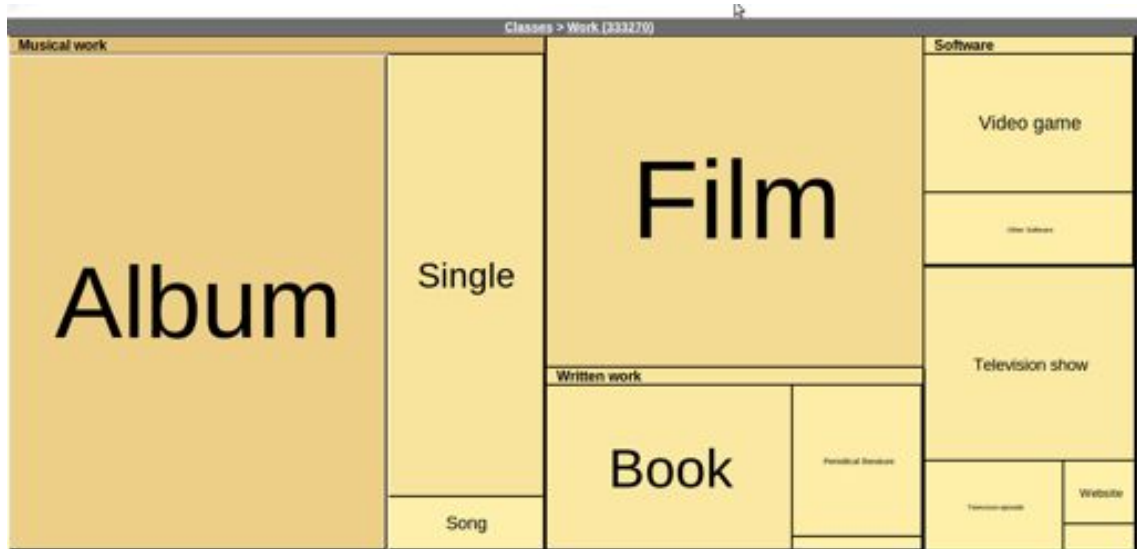


Figure 7.16: Treemap of the DBpedia dataset (2)

7.3.3.1. Experimental Design

The tests were conducted at the UsabiliLAB²⁹ where sessions were registered using Morae Recorder together with Morae Observer³⁰ to analyse test data. The software collected the task time and the effectiveness (completed, not completed and partially completed). Users were presented Rhizomer with the DBpedia dataset and the four components to use. They were encouraged to use the think aloud protocol [Lew82] to express their opinion and thoughts on the system while performing tasks.

We asked 10 participants to participate in the study (6 males, 4 females), having a unique profile characterized by good knowledge of information technology, they work regularly with computers and use internet. None of them had expertise in Semantic Web technologies or had previously used the DBpedia.

7.3.3.2. Tasks

The chosen tasks comprise the common information needs when users navigate and want to find something in a web site [RM02]: known-item seeking, exhaustive research, re-finding. The items to find slightly vary for each task and component to reduce the effect of learning:

1. **Known-item seeking:** find one node for different levels of the hierarchy using navigation menus (1), site map (2), treemap (3) and site index (4).
 - **Task 1:** find 1st level item - Work (1), Place (2), Event (3), Species (4).
 - **Task 2:** find 2nd level item - Monument (1), Insect (2), Aircraft (3), Film festival (4).

²⁹<http://griho.udl.cat/about/infrastructure.html>

³⁰<http://www.techsmith.com/morae.html>

- **Task 3:** find 3rd level item - Reptile (1), Sports team (2), Hospital (3), Golf Player (4).

2. Exhaustive research:

- **Task 4:** comparing node size. Arrange from bigger to smaller in number of resources using navigation menus (1), site map (2) and treemap (3).
 - Place, Agent and Species (1).
 - Work, Place and Mean of transportation (2).
 - Eukaryote, Person and Musical Work (3).
- **Task 5:** topology understanding. Using navigation menus (1), site map (2) and treemap (3) find subtypes of:
 - Mean of transportation (1), Species (2), Work (3).

3. Re-finding: find previously visited items using navigation menus (1), site map (2), treemap (3) and site index (4).

- **Task 6:** find Monument (1), Insect (2), Aircraft (3), Film festival (4).

In total, users had to perform 6 tasks using navigation menus, 6 with the site map, 6 with the treemap and 4 using the site index. 12 tasks belong to known-item seeking, 6 to exhaustive research and 4 to re-finding. We ran the evaluation as a within-subject design, where each participant performed all tasks using all four components, except for tasks 4 and 5, which were not performed using the site index because it does not provide the context information needed to perform these kind of tasks. Users were given a maximum of 2 minutes to perform each task.

7.3.3.3. Usability metrics

In this evaluation we have only considered some of the metrics proposed in SWET-QUM. Other metrics are more appropriate for exploratory tasks, which users can complete through different ways. In this case, users must perform short tasks with specific components. For the usability test we have chosen the following metrics:

- Effectiveness:
 - Task success.
- Efficiency:
 - Task time.
 - Number of help requests.
- Satisfaction:
 - Satisfaction questionnaire.

7.3.3.4. Results and Discussion

Overall, users were able to complete most tasks without problems. The task success was 100% for all tasks except for task 3 using the site map, with only 20%. This task was the only one that users had problems to complete, with 8 participants asking the facilitator for help. The issues related with this task are explained later.

Figure 7.17 shows the average task completion times. In task 3 with the site map we have only considered the task time for those users who were able to complete the task. The results show that user's performance depended on a combination of the task and the component used. The results are discussed next:

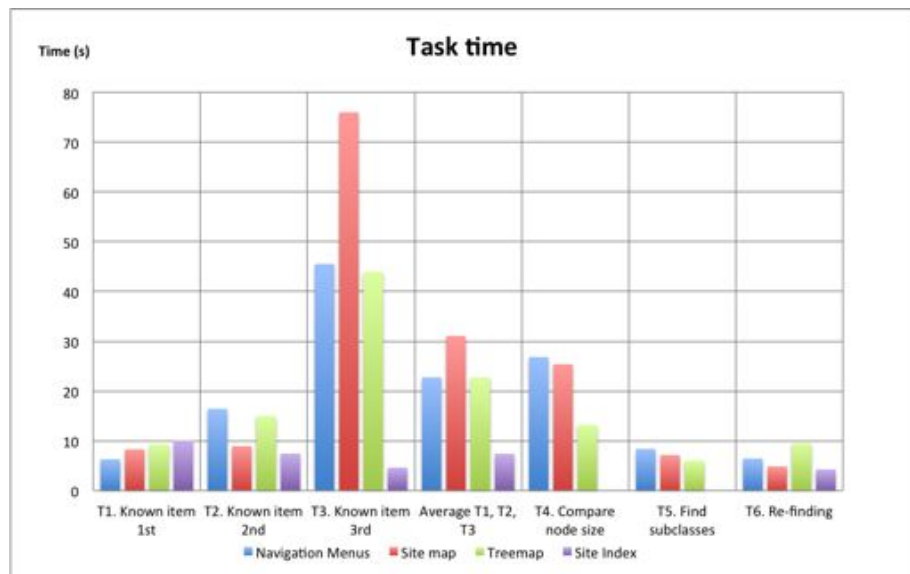


Figure 7.17: Average task completion times

Known-item finding

Navigation menus were the fastest component to find large known-items located in the first level of the hierarchy (e.g. Work). Users could see the most important items at a single glance in a horizontal line. However, their efficiency decreases when users have to find items located deeper in the hierarchy (e.g. Monument or Reptile).

With the site map, users could easily find nodes located in the first or second level of the hierarchy (e.g. Place or Insect). However, the *summarized* site map only shows two levels of the hierarchy and users had to switch to the *full* site map to find items located deeper (e.g. Sports team). Only 2 users were able to find this option and were able to complete the task. In any case, they needed a lot of time to change to the full site map.

Something similar occurs with the treemap. Users could easily find large nodes located in the first or second level of the hierarchy (e.g. Event) but it was more difficult for them to find smaller nodes (e.g. Aircraft) because their label was not visible until they zoomed-in.

It was difficult for users to find nodes located deep in the hierarchy (e.g. Aircraft, Hospital) using the treemap, site map or navigation menus. They needed to understand how the dataset was hierarchically structured and guess where the item could be located. In these cases, the site index was the fastest component because users did not need to understand the site structure. They only needed to locate the item in the alphabetical index.

Exhaustive research

The treemap was the fastest component to compare node sizes. Rectangle areas allowed users to easily compare different nodes. On the other hand, using navigation menus or the site map users had to manually compare the number of instances. Several users pointed out that it was difficult because thousands were not separated by dots. The treemap was also the best component to understand the topology and to find a class and its subclasses. Different colors allow to easily identify groups of related nodes. However, in this case there is not a big difference compared to navigation menus and the site map.

Re-finding

Users found previously visited nodes (T6) faster than the first time (T2) with all components. The site map was significant faster than navigation menus. The treemap was the worst component to perform this task. The visual representation of nodes did not help much users to remember where the item to find was located.

User satisfaction

After finishing the tasks with each component, we gave users a written questionnaire to evaluate their confidence, satisfaction or frustration with it. Users were asked to rate the following statements from 1 to 7, being 1 the most negative and 7 the most positive. Table 7.7 shows the questions and user ratings.

| Question | NM | SM | TR | SI |
|---|------|------|------|------|
| Q1. The information showed on this component is helpful | 6.2 | 6.2 | 5.7 | 6.4 |
| Q2. This component is easy to use | 6.4 | 6.2 | 5.7 | 6.6 |
| Q3. Using this component, it was easy to find the information I was looking for | 6.3 | 5.3 | 5.5 | 7 |
| Q4. From this component, I understand the structure of the website | 6.3 | 6.6 | 5.4 | 4.6 |
| Q5. From this component, I understand what content is available on the website | 6.2 | 6.3 | 4.9 | 5.8 |
| Average | 6.28 | 6.12 | 5.44 | 6.08 |

Table 7.7: User satisfaction questionnaire for navigation menus (NM), site map (SM), treemap (TR) and site index (SI)

Overall, users had a good impression of the overview components. Users prefer navigation menus, the site map and the site index instead of the treemap. They are used to common web components more than to interactive visualization methods. Most of them had never seen a treemap before and required some time to learn how to use it. There were no significant differences between traditional web components in terms of use preference. It is important to notice that the site index was considered the best component to find concrete information (Q3) but the worst to understand the site structure (Q4).

7.3.3.5. Conclusions and proposals

We presented four components that support the overview task proposed by Shneiderman in his visual seeking mantra. These components are useful for obtaining a broad view of the datasets, their main types and the relationships between them. The RDF data model being prevalent on the Data Web enables us to create this overview in a generic and automatic way. A good overview provides users an appreciation of the collection of objects and can help achieving their information seeking goals. Rhizomer, with these overview components, allows users to explore datasets even if the publisher of the data does not provide any exploration or visualization means.

Our evaluation shows that each component has its strengths and weaknesses depending on the use case. Navigation menus are useful to find the most important classes but they can only show a limited number of options. Site maps provide a general top-down view of the overall site contents but users need to understand the site structure to find concrete items. Treemaps are useful to get an overview of the data, compare node sizes and identify groups of related items through different colors. However, it is very difficult to locate those nodes that have a small area. A site index provides easy access to particular content but does not give information about how a website is structured.

From test results and their analysis, we have elaborated these proposals to improve the existing components:

Navigation menus

The proposal for navigation menus is to improve the algorithm that generates them. Right now, the only factor considered to generate the menus is the number of instances of each class. As a result, it is possible that some options in the first level of the generated menu do not have subclasses. For example, in our evaluation with DBpedia, the class `Person function` appeared in the first level but did not have any subclasses. Then, we can not use a submenu for this class and we waste space in the navigation menu that could be used to show more options. Meanwhile, other classes with less instances but with more subclasses do not appear in the first level. The proposal is to study how to generate the navigation menus considering both factors: the number of instances and the number of subclasses for each class.

Site map

As the results show, users did not understand the difference between the two site map versions and did not know how to switch between them. The first one reflected the original hierarchy of the dataset, displayed as a tree with multiple levels. The second one, related with the structure of navigation menus, was a summarized version with the main classes. Only two users were able to find how to switch between both versions.

In other studies regarding site map usability [Ted08], participants successfully used site maps that were not organized to reflect the site's main navigation. Moreover, these studies suggest that multi-column site maps work better because users need less scrolling to get an overview of the site's structure. Therefore, our proposal is to keep only the full site map and display it using two columns. The full version allows to reach all the classes in the original hierarchy while the summarized version only shows the main classes.

Treemap

The main issue with treemap was to find small nodes. In some cases, there was not enough space to show their labels inside small rectangles. Users could only see these labels when they moved the mouse pointer over the rectangles. Our proposal is to study how to group small nodes into a supernode. Then, when zooming into this supernode, it would be easier to locate small nodes. We also propose to complement the treemap with a keyword search that could allow to find these nodes.

7.4. Iteration 4

7.4.1. Requirements analysis

The main issues spotted during the evaluation in iteration 2 (section 7.2) are related with the pivoting operation. First of all, users did not pay attention to the interface components providing pivoting. Another problem that hinders usability is the lack of context between pivoting steps. For example, some users saw pivoting as restarting exploration for a new class of resources. Many interaction challenges need to be investigated in order to mitigate these usability issues.

Finally, regarding our evaluation in iteration 3 (section 7.3), we identified an important issue related with navigation systems. Some users prefer to navigate step by step in the hierarchy instead of trying to find the target class directly. In these cases, local navigation should allow users to navigate to subclasses.

We identified the following formal requirements for this iteration:

- **Encourage pivoting:** the elements providing pivoting should be more relevant in the user interface to encourage users to use them. Related with challenge 4.
- **Improve user contextualization between pivoting steps:** users should be able to see the pivoting path they have taken and go back to previously visited pages to follow a different path. Related with challenge 5.
- **Improve local navigation:** local navigation should be improved to allow users to navigate to subclasses of the current class viewed. Related with challenges 1 and 2.
- **Sorting and paginating results:** when a query returns to many results, it is necessary a method to sort and paginate them. Related with challenges 4 and 7.
- **Reduce interaction steps:** the user interface should be re-designed in order to reduce the number of interaction steps necessary to perform tasks. Related with challenge 7.

7.4.2. Design and implementation

7.4.2.1. Facets re-design

An issue detected in previous iterations was the number of steps necessary to perform tasks in our faceted browser. This is due to the fact that users had to unfold each facet to be able to search on it. Moreover, the SPARQL query to obtain the 5 most common values for a facet was always executed, even if users used the autocomplete search box.

We have redesigned facets user interface to reduce the number of interaction steps and avoid executing unnecessary SPARQL queries. By default, each facet is displayed as an autocomplete search box and a button to unfold the five most common values. In this way, users can easily search on each facet without expanding it, reducing the number of necessary interaction steps.

7.4.2.2. Encouraging pivoting

Every visual component related with pivoting should be more prominent in the user interface. All elements should encourage pivoting and provide context information to users. To improve pivoting, the following components have been redesigned or proposed:

- Connections: the right side of the user interface shows a list of links to classes for which it is possible to pivot to from the current faceted view. Those classes in which the user has already pivoted are listed in a separate box named “Active connections”. In this way, users can easily identify possible navigation paths and the pivot operation looks like following common links to other pages. This component is now more prominent in the UI compared to our previous design.
- Inverse properties in resource descriptions: when users see details of a concrete resource, they can also pivot to a set of related resources through inverse properties. For example, a user who reaches the page describing “Woody Allen” in LinkedMDB, can pivot to the list of films he has directed through the inverse property “Is director of film”.
- Tooltips: to encourage pivoting we added tooltips to every element that produces this functionality. As a result, when the mouse pointer moves over these elements, a message is displayed to users explaining the action that will be performed if they click: “Switch to related Film”.

Figure 7.18 shows the components implemented and redesigned to encourage pivoting. It also shows the new design for facets.



Figure 7.18: Improvements in pivoting in iteration 4.

7.4.2.3. New breadcrumbs design

Up to now, we have only implemented attribute breadcrumbs, which show selected facet values and allow users to remove active filters. Our proposal for this iteration is to redesign breadcrumbs to improve context information implementing the other types of breadcrumbs identified by Instone [Ins]: location and path breadcrumbs.

However, we believe that displaying three different types of breadcrumbs in the same page can be confusing for users. Therefore, location and path breadcrumbs are displayed together in the same area, separated from attribute breadcrumbs. Location breadcrumbs show where the page is located in the class hierarchy. They are static and do not change during the user exploration of that class. Path breadcrumbs are concatenated with location breadcrumbs, showing the executed pivoting steps.

Moreover, we have also integrated look-ahead breadcrumbs [BAI05] as part of location breadcrumbs to provide local navigation. Regular breadcrumbs show the trail of links leading to the current resource, while look-ahead breadcrumbs can improve the navigation by including a list of links to resources that are reachable from that particular page. In our case, each element in location breadcrumbs shows a drop-down list with links to its subclasses. A study of look-ahead breadcrumbs suggest that they can lead to more efficient site navigation and improve the user experience [BAI05].

Figure 7.19 shows a screenshot of our experimental breadcrumbs, including location, path, attribute and look-ahead breadcrumbs.

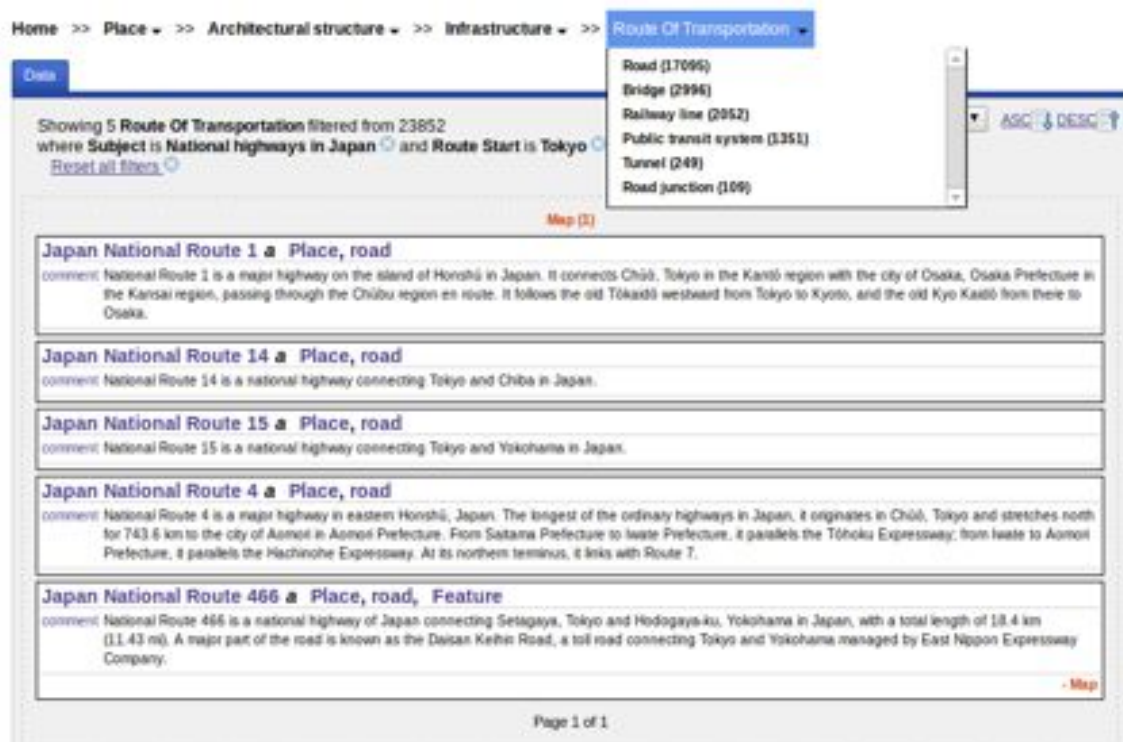


Figure 7.19: Experimental breadcrumbs including location, path and attributes.

```
1 SELECT DISTINCT ?r1
2 WHERE {
3   ?r1 rdf:type <http://data.linkedmdb.org/resource/movie/film> .
4   ?r1 <http://purl.org/dc/terms/date> ?sort
5 }
6 ORDER BY ASC(?sort)
7 OFFSET 10
8 LIMIT 10
```

Example 24: SPARQL query after sorting by date and paginating

7.4.2.4. Paginating and ordering results

We have implemented two other components to complement faceted browsing. The first one is a method to paginate results. When a SPARQL query returns too many results to be displayed, pagination is used to show only a subset of the result. Pagination is supported by most SPARQL endpoints by using the LIMIT and OFFSET modifiers.

The second component is a widget to sort results according to some criteria. Those properties whose values are ordinal data can be used to sort results. We recognize these properties by their range, when it belongs to suitable data types from XML Schema, e.g. integer, float, date, etc.

Figure 7.20 shows a screenshot of both components and Example 24 the SPARQL query generated after sorting the results by date and setting a pagination of 10 elements.



Figure 7.20: Pagination and sorting components

7.4.3. Evaluation

7.4.3.1. Experimental Design

The tests were performed in the UsabiliLAB³¹ facilities at Universitat de Lleida. We used two computers, the first one equipped with an Eye-Tracker and Morae Recorder to register user interaction, and the other one equipped with Morae Observer and Morae Manager to observe and analyse the session.

14 users participated in the test, all of them considered in the lay-user profile. We defined 2 groups: group A with users who were new to the system and group B with users who had previously interacted with *Rhizomer*. In both cases, the groups involved 7 users, being more than 5 users as recommended by Nielsen [NL94] for qualitative user tests.

We also wanted to compare the navigation systems offered by *Rhizomer* with *Visor* and *gFacet*, the remaining tools that also provide functionalities similar to pivoting. However, the performance of both tools is very poor, particularly with large datasets as the ones used in the evaluation. That makes impossible to perform the same tasks in reasonable time.

The objectives of the evaluation were:

- To determine whether the user interface is balanced and user's attention is better distributed among the relevant components, especially among those that provide pivoting.
- To validate that the new design and the introduction of new components solve the problems identified in previous evaluations.
- To compare the two groups defined and see whether users who are not new to the system remember how to use it.
- To detect other usability problems and obtain spontaneous feedback from the users.

7.4.3.2. Tasks

We decided to keep the same tasks from the evaluation in Iteration 2 in order to validate if there was improvement with the new user interface design and components. For group B, with users who had already interacted with *Rhizomer*, the tasks were slightly different, being also possible to compare them. As in our second evaluation, the first task was introductory to the system and we did not measure the metrics for it. The complete set of tasks was:

- **Group A:**
 - **Task 1A:** find 5 films starring "Orlando Bloom".
 - **Task 2A:** find 5 films with "Woody Allen" both as director and actor.
 - **Task 3A:** find directors of films in countries located in "Oceania".
- **Group B**

³¹<http://griho.udl.cat/about/infrastructure.html>

- **Task 1B:** find 5 films starring “Bruce Willis”.
- **Task 2B:** find 5 films with “Woody Allen” both as writer and director.
- **Task 3B:** find directors of films in countries that have the currency “Euro”.

7.4.3.3. Usability metrics

As in our second iteration, the evaluation was based on SWET-QUM and we chose the following metrics:

- Effectiveness:
 - Task success.
 - UI Component effectiveness.
- Efficiency:
 - Task time.
 - Number of help requests.
 - Task efficiency: percentage of task success per time.
 - UI Component Efficiency.
- Context coverage:
 - Task flexibility.
 - Layout flexibility.
- Satisfaction.

7.4.3.4. Results and discussion

Effectiveness and Efficiency

Table 7.8 shows the results comparing the tasks with our previous evaluation of *Rhizomer* in iteration 2.

In group A, 85% of users were able to complete task 2A without help and only one user needed assistance to complete the task. In task 3A, 28% of users were able to complete the task without help. Including assistance, 71% of users were able to complete the task. For group B, with users who had previously interacted with *Rhizomer*, all users were able to complete task 2B without asking the facilitator for help. In task 3B, 57% of users completed the task without help. Including those users who asked for help, the task success in task 3B is 100%.

From the point of view of efficiency, the results do not show an improvement compared to our previous evaluation. The average time needed to perform both tasks for the two groups was higher than in iteration 2. This could be due to the following reasons. First of all, the user interface incorporates new components, which reduces its simplicity. Although these components allow users to perform complex tasks, they need more time to identify which components to use. This hypothesis is also supported by the Task Success metric. The task success has increased in complex tasks such as 3A and 3B, but they required more time to be completed.

7.4. ITERATION 4

| Metric | | Task 2 Iteration 2 | Task 2A Iteration 4 | Task 2B Iteration 4 | Task 3 Iteration 2 | Task 3A Iteration 4 | Task 3B Iteration 4 |
|------------------|-------------------------|-----------------------|------------------------|------------------------|-----------------------|------------------------|------------------------|
| Task time (m) | Minimum | 0.89 | 2.90 | 1.17 | 1.99 | 4.47 | 2.33 |
| | Maximum | 2.23 | 5.28 | 5.33 | 4.50 | 7.78 | 5.12 |
| | Mean | 1.69 | 3.66 | 2.31 | 3.43 | 5.87 | 3.90 |
| | Std. Dev. | 0.57 | 0.94 | 1.52 | 0.96 | 1.74 | 1.15 |
| Task success | Without assistance | 100% | 85% | 100% | 0% | 28% | 57% |
| | Including assistance | 100% | 100% | 100% | 100% | 71% | 100% |
| Task efficiency | | 68% | 23% | 43% | 0% | 5% | 15% |
| Help requests | | 0 | 1 | 0 | 7 | 5 | 3 |

Table 7.8: Comparison with iteration 2

The second reason could be that the selected user profile is too broad. We have considered as lay users all those participants who do not know about Semantic Web technologies. However, this profile can include users who work with information technologies (without knowledge about Semantic Web technologies) but also users who only use computers and internet as a support tool or in their leisure time. For example, thanks to attribute breadcrumbs, a user expert in SQL databases was able to identify pivoting as “an operation similar to JOIN in SQL”. This analogy allowed him to understand this operation and finish the proposed tasks. On the other hand, users with less IT expertise found it difficult to understand pivoting because it is not a common operation for them.

Overall, users from group B had more success and needed less time to complete tasks than users from group A. This is related with the fact that they had previously interacted with the system and most of them could remember how it worked, despite the changes in the UI. The most promising outcome is in the number of help requests. It is important to highlight that some users were able to complete tasks 3A and 3B without help, while in the previous iteration, for the same task, all users required facilitator assistance. In iteration 2, most users asked for help because they did not understand the pivoting operation. The new breadcrumbs provide more context information and show the pivoting steps performed, which help users to understand this operation.

UI Component Effectiveness and Efficiency

Table 7.9 presents the values for the UI component effectiveness and efficiency metrics. For both groups, the Data Exploration UI Effectiveness is 100% because all relevant UI components for data exploration did receive some attention by users during the evaluations. However, it is important to highlight that when considering UI Components Efficiency, there are components that received really little attention (highlighted with light grey for each tool) while others received a lot (highlighted with dark grey).

| Relevant UI Components | UI Component Efficiency | | Data Exploration UI Effectiveness | |
|---------------------------|-------------------------|---------|-----------------------------------|---------|
| | Group A | Group B | Group A | Group B |
| Global Navigation Bar | 8.5% | 9.1% | 100% | 100% |
| Facets | 45.5% | 42.5% | | |
| Connections | 9.6% | 11.6% | | |
| Attribute breadcrumbs | 11% | 15.4% | | |
| Location+Path breadcrumbs | 2.9% | 2.5% | | |
| Results | 22.4% | 18.9% | | |

Table 7.9: Evaluation results for iteration 4: UI component effectiveness and efficiency

As in the evaluation in iteration 2, facets were the component that received most of the attention. However, the attention has been reduced from 70% in iteration 2 to 45.5% - 42.5%. In this case, some of the attention is captured by the results list, from which it was also possible to pivot through properties, e.g. from woody allen to the related films he has directed. Some users performed pivoting in this way.

Breadcrumbs received also more attention, increasing from 11% to 13.9% - 17.9% considering location, path and attribute breadcrumbs together. They were more prominent in the UI and users could identify them.

Figures 7.21 and 7.22 show the heat maps of *Rhizomer* for both user groups defined. They illustrate the data from the eye-tracker used to compute the UI Component Efficiency metric. Overall, user attention was better distributed across the user interface than in iteration 2. As it can be seen, there are not much differences between both groups. However, while users in both groups were able to see the connections box, most users in group A did not understand its meaning. On the other hand, users in group B knew its functionality and used it to perform pivoting.



Figure 7.21: Rhizomer heat map, Iteration 4, user group A

7.4. ITERATION 4



Figure 7.22: Rhizomer heat map, Iteration 4, user group B

Context coverage

The Task Flexibility and Layout Flexibility metrics were measured to determine the context coverage. Table 7.10 shows the measures for both metrics, highlighting in grey the improvements compared to iteration 2.

For Task 2A Flexibility, it was determined that, from the conceptual point of view, users should be able to perform the task following 3 main paths: starting from actor, starting from director or starting from film. Similarly, users should be able to perform Task 2B from 3 main paths: starting from director, starting from writer or starting from film. For Task 3A Flexibility, users should be able to complete the task following 4 main paths: starting from director, film, country or continent. Similarly, users should be able to perform Task 3B following 4 main paths: starting from director, film, country or currency.

Compared to our previous evaluation, the Task Flexibility for Task 3A and Task 3B has increased. In *Rhizomer's* second evaluation, these tasks could only be completed starting from film or director because country was not available from navigation menus. Now it is also possible to start from country thanks to the sitemap, treemap or site index.

Moreover, the number of interaction steps necessary to perform Task 2A and Task 2B has been reduced. In our previous evaluations, users had to expand each facet before being able to search on it. Now, the search box for each facet is directly available without needing to unflod it. As a result, the Layout Flexibility metric shows a better value than in previous evaluations.

Satisfaction

Users filled in the same questionnaires as in the second iteration in order to measure their satisfaction with *Rhizomer*. In the first questionnaire, presented after completing each task, they had to rate from 1 to 5 these particular questions:

| Group | Metric | Task 2 | | Task 3 | |
|---------|--------------------|--------|---|--------|--|
| Group A | Task Flexibility | 100% | It is possible to go through actor, director or film | 75% | It is possible to go through director and film from the navigation menu. Country is also available from the site map, treemap or the site index. Continent is not available. |
| | Layout Flexibility | 1.2 | Zoom film (0), search box actor (1), select person (2), search box director (1), select person (2) | 1.83 | Zoom film (0), pivot country (1), expand continent (2), select Oceania (3), pivot film (2), expand director (3) |
| | Interaction Steps | 5 | | 6 | |
| Group B | Task Flexibility | 100% | It is possible to go through writer, director or film | 75% | It is possible to go through director and film from the navigation menu. Country is also available from the site map, treemap or the site index. Currency is not available. |
| | Layout Flexibility | 1.2 | Zoom film (0), search box director (1), select person (2), search box writer (1), select person (2) | 1.83 | Zoom film (0), pivot country (1), expand currency (2), select Euro (3), pivot film (2), expand director (3) |
| | Interaction Steps | 5 | | 6 | |

Table 7.10: Evaluation results for the iteration 4: context coverage

- **TA1** The task was... (1) very hard - (5) very easy.
- **TA2** I think that I have done the task... (1) not correctly at all - (5) absolutely correct.
- **TA3** The interface structure... (1) did not help me at all - (5) did help me very much.
- **TA4** The time to complete task has been... (1) long - (5) short.
- **TA5** To achieve the task I have had to be... (1) very focused - (5) not focused at all.
- **TA6** The task was... (1) badly defined, I did not understand the objective - (5) well defined, I understood the objective.

Additionally, the following questionnaire was presented to users after they had completed all tasks:

- **TE1** It is easy to use the tool - (1) disagree ... (5) agree.
- **TE2** The system is intuitive - (1) disagree ... (5) agree.
- **TE3** I had fun using it - (1) disagree ... (5) agree.
- **TE4** The options are easily identifiable - (1) disagree ... (5) agree.

The results for the post-task satisfaction questionnaires are shown in Figure 7.23 and Figure 7.24, which compare the post-task satisfaction for both tasks and both groups with iteration 2. As it can be observed, the satisfaction measures for both tasks are better for group B, with users who had previously interacted with *Rhizomer*. There seems to be a learnability effect that explains the increased satisfaction. The bad results in group A are mainly due to the fact that some users were not able to complete the tasks. Finally, it is important to notice that both groups considered Task 3 as hard and they needed to be very focused to perform it.

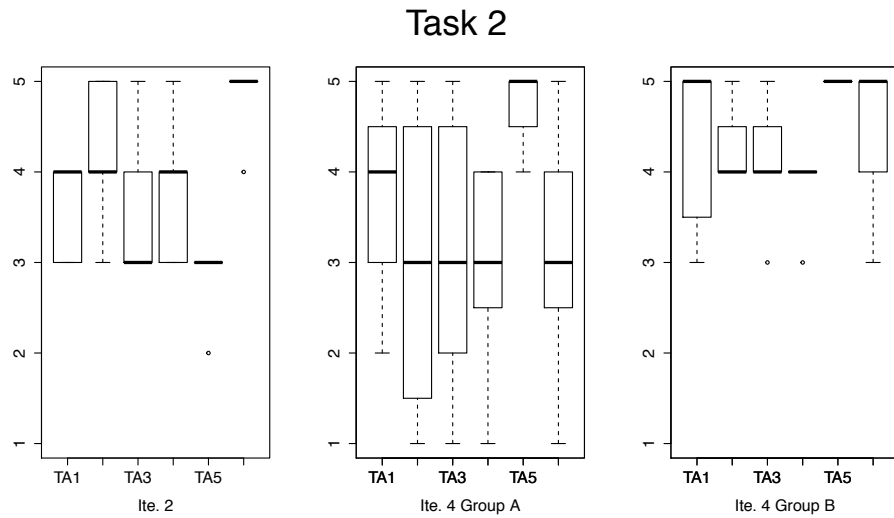


Figure 7.23: Post task satisfaction measures for iteration 4, task 2

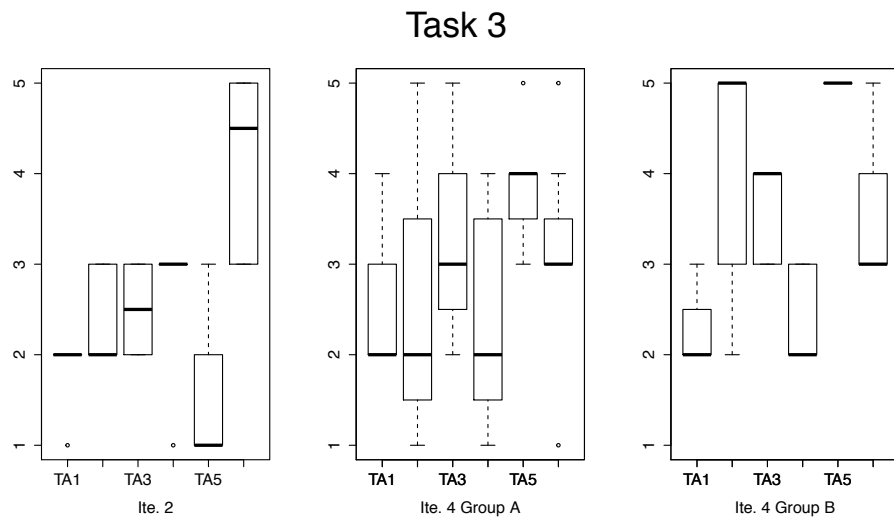


Figure 7.24: Post task satisfaction measures for iteration 4, task 3

After completing all tasks and post-task questionnaires, users also filled a post-test questionnaire that tried to capture their overall satisfaction and user experience with *Rhizomer*. The results are shown in Figure 7.25. As it can be observed, the satisfaction measures for group B are the best ones, probably because users were already familiar with the system. The results for group A are similar to the ones obtained in iteration 2.

7.4.3.5. Conclusions and proposals

Our evaluation results show that the new design helped users to better understand the user interface and its functionalities. With the new design, the user interface is more balanced and it is easier to perform and understand pivoting. Contrary to the tests in iteration 2, some users were able to complete task 3 without receiving any help.

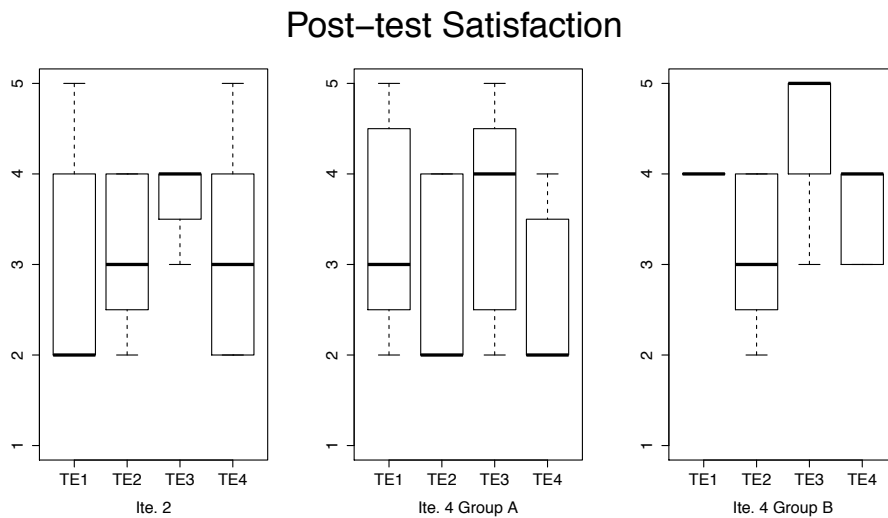


Figure 7.25: Post test satisfaction measures for iteration 4

However, from test results and their analysis, some issues were identified and these proposals were elaborated to improve the system:

- While users in group B knew the relevance of the connections box for pivoting, some users in group A did not pay much attention to it. They said that they did not look at the right side because it normally contains non-significant content. Moreover, some users mentioned that they are used to see ads in that place and the connections box looked like Google Ads. Therefore, we believe that this component should be re-designed.
- Each resource from the result list shows its label and the classes it belongs to, e.g. “Woody Allen, a person, director”. In this case, `director` is a subclass of `person`. That was difficult to understand for some users. For example, a participant said: “Can I choose Woody Allen as director or person? What does it mean?”. The proposal to solve this issue is to show only the most specific class for each resource.
- Some participants had problems with the vocabulary used to describe this dataset, especially in tasks 3A and 3B. For example, some users tried to perform the task by filtering the property `Featured Film Location` instead of the property `Country`. However, this property links to cities but not to countries, which makes impossible to perform the task. Other users were confused between countries and continents. It took some time for them to realize that “Oceania” was a continent and not a country. However, it is important to notice that tasks 3A and 3B were complex and difficult to perform.
- Although users new to the system were able to perform some pivoting steps, most of them were still confused about this operation and could not finish task 3. However, they were taught how to perform this task after the test and then they could understand pivoting. Our proposal is to add a tutorial or demonstration video that could help users understand how the system works.

7.5. Iteration 5

7.5.1. Requirements analysis

While the results so far support our hypotheses, the evaluations also point some aspects that let us rethink the design of the user interface. The developed components allow users to create complex queries without requiring Semantic Web expertise. However, most users typically try to find information not through advanced exploration but through more simple means. In fact, they usually dump initial thoughts in form of keywords in a search box and glance through the most relevant results until they are satisfied.

After these evaluation rounds we could observe how users try to perform the proposed tasks. When users are proposed a task, they tend to think about concrete things instead of collections of elements. Some of the proposed tasks were to find films with certain characteristics: a concrete director, starring an actor, etc. For example: “Find movies starring Bruce Willis” or “Find movies where Woody Allen is both actor and director”. These tasks are easy to perform if users focus on film and they filter the appropriate facets. However, users’ attention focuses on Bruce Willis or Woody Allen. Something similar occurs with the task “Find directors of films in countries located in Oceania”. Instead of thinking about directors or films, users focus on Oceania. They try to reach the page describing that concrete resource and then navigate to related resources from it.

Based on these experiences and the test results, a search component has been designed and integrated in the User Interface. Another proposal is to implement facet widgets for concrete datatypes such as numbers, dates, etc. Finally, we also propose to apply visual transitions in pivoting to improve context information. In concrete, we derived the following requirements for this iteration:

- **Support keyword search:** users must be able to find specific resources or classes through keyword search. These resources can serve users as a starting point for further navigation. Related with challenge 1.
- **Advanced widgets for facets:** each facet should be adapted to the characteristics of its values, e.g. nominal data, ordinal data, geospatial data, etc. This would allow users to select ranges instead of single values. Related with challenge 4.
- **Visual transitions in pivoting:** the pivoting operation is still confusing for some users. Visual transitions can be helpful for users to understand this operation. Related with challenge 5.

It is important to notice that these components are still under development and have not been evaluated yet. Therefore, this iteration should be considered as future work.

7.5.2. Design and implementation

7.5.2.1. Keyword search

Keyword search is a well established technique in most websites and used by a large number of users [Hen07]. Research has shown that keyword search and facet browsing complement each other [WKS10]. Both interaction styles should be supported simultaneously we need a method for combining them. The proposal is to implement a keyword search to provide users starting points to explore data. Through keyword search, users can have easy access to particular known content.

Figure 7.26 shows the proposed search system, which is divided into three components:

- A search box, where users type the keywords. It includes an autocomplete widget [W06] that shows resources lexically related with the specified keywords .
- A main area showing the list of result that match the specified keywords.
- A secondary facet area where users can filter results by their type.

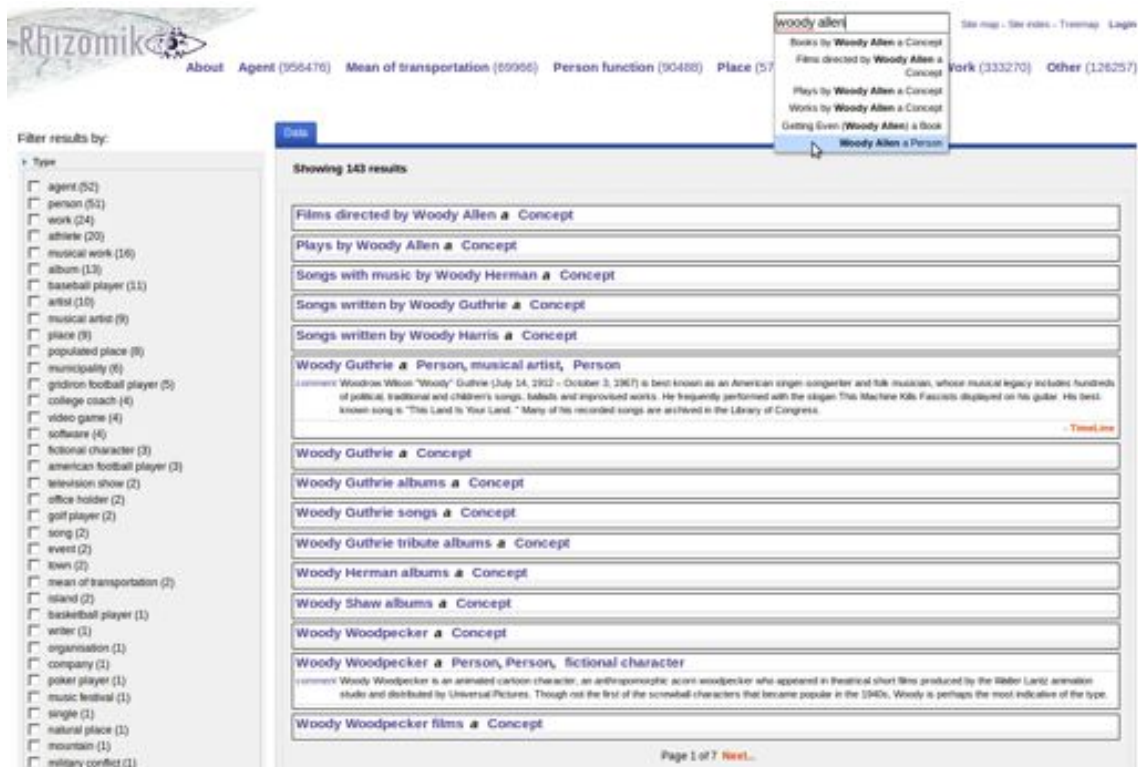


Figure 7.26: Keyword search with results, autocomplete widget and type filtering

7.5. ITERATION 5

Keyword search provides users a new interaction model. In addition to selecting a class from navigation menus or other overview systems, users' initial step can be a keyword search to locate specific resources. Then, they can refine their search by choosing the type of resources they are looking for. Once users have selected a type, they are presented the list of facets for that type of resources, as detailed in our previous iterations. From here, they can continue exploring data by filtering other facets or pivoting to related resources.

Another possibility for users is to select a resource of interest and continue navigating from it. It is possible to navigate to a particular related resource or pivot to a set of related resources. In fact, that is how most users tried to perform the tasks in our evaluations. Figure 7.27 shows the page describing "Woody Allen". From here, it is possible to see properties that describe this resource and pivot to related resources.

The proposed keyword search system is simple and only retrieves resources that match the specified keywords. However, keyword search over RDF graphs is a wide research area. Topics such as mapping natural language to SPARQL queries [SAN13] or entity ranking [BMV11] should also be considered.

The screenshot shows a web interface for a resource description. At the top, there is a blue header with the word "Data". Below it, a red "Timeline (1)" indicator is visible. The main content area is titled "Woody Allen a person, Concept, Thing, agent". It contains an abstract paragraph about Woody Allen's career, followed by a list of properties and their values. The properties include "active years start year", "ALTERNATIVE NAMES", "birth date", "birth name", "birth place", "birth year", "box width", "Caption", "child", and "is director of film". At the bottom of the page, there is a red box containing the text "See related films" and "Pivot to directed films".

Figure 7.27: Resource describing Woody Allen and pivoting options

7.5.2.2. Facet widgets

Most of the current browsers use the same widgets for all facets, regardless of the characteristics of each facet. The standard way to display facets is a list where facet values can be selected. However, depending on the kind of data, some UI mechanisms are more suitable for those facets than others.

For example, ordinal data can be restricted by intervals since they follow a certain order. In these cases, choosing a single value is often useless. For some facets, such as price or dates, users may be interested in defining a range of values between a maximum and a minimum. Then, a slider widget can be used to define ranges, e.g. values from 10 to 20. Facets with a hierarchical topology can be displayed by the classical tree directory with subvalues. Geospatial 2D data can be displayed in a map and select an area of interest.

It is possible to identify such facets when they are described using common vocabularies or when their range belongs to datatypes from XML Schema³². Our proposal is to display special widgets for these ranges as detailed in Table 7.11

| RDF Vocabulary | Data type | Widget |
|---|--------------|-----------------------------|
| xsd:int xsd:decimal xsd:float xsd:double | Ordinal | Slider |
| xsd:date ical:dstart | Temporal | Calendar slider |
| wsg84:lat / wsg84:lon vcard:latitude / vcard:longitude georss:point | Geospatial | Map area selector |
| rdfs:subClassOf skos:narrower | Hierarchical | Tree with subvalues |
| xsd:string | Nominal | List with choices (default) |

Table 7.11: Special facet widgets

Figure 7.28 shows two widgets implemented for ordinal and temporal facets. Ordinal facets are displayed with a slider, which allows to define ranges between two numbers. Temporal facets are displayed in a calendar, allowing to select a range between two dates.

7.5.2.3. Transitions in pivoting

Whenever pivoting occurs, the user interface changes accordingly: a new result set is shown in the centre of the screen, the lists of facets and connections are updated and a new element appears in the breadcrumbs showing the new pivoting step. Updating every control on the screen at the same time places a substantial cognitive load on users. They have to keep track of all these changes.

When the user interface is divided into multiple components, visual transitions can help users to maintain their contextualization [HR07]. Our proposal is to explore CSS3 and Javascript transitions to gradually update the components on the screen. In this way, users should be able to better understand pivot operations.

³²<http://www.w3.org/TR/xmlschema11-2/#built-in-datatypes>

7.5. ITERATION 5

The screenshot displays a web application interface for filtering a list of films. On the left, a sidebar titled "Filter Film by:" contains several filter categories, each with a search input and a "Show values" button:

- Page:** Search Resource... Filter Resource
- Label:** Search... Show values
- Title:** Search... Show values
- Filmid:** From 1 To 96816
- Performance:** Search Performance... Filter Performance
- Actor:** Search Actor... Filter Actor
- Date:** From 0 To 2013
- Person (in Made):** Search Person... Common values
- Director:** Search Director... Filter Director

The main content area shows a list of film titles, each with a small "a" icon and the word "film":

- Airplane! a film
- American Beauty a film
- Anywhere But Here a film
- Buffy the Vampire Slayer a film
- Grand Hotel a film
- Ice Age a film
- O Brother, Where Art Thou? a film
- Stargate a film
- The War of the Roses a film
- Wild Strawberries a film

At the top of the main content area, there is a navigation bar with "Home >> Film" and tabs for "Data" and "Charts". Below the navigation bar, it says "Showing 85620 Film filtered from 85620" and "Sort by: A-Z" with "ASC" and "DESC" options. A "Reset all filters" link is also present. At the bottom of the main content area, it says "Page 1 of 8562 Next..." and a "WS" logo is visible.

A calendar widget is overlaid on the "Date" filter, showing the month of September 2013. The calendar has a header with "Su Mo Tu We Th Fr Sa" and a grid of dates from 1 to 30. The date 27 is highlighted with a mouse cursor.

Figure 7.28: Facet widgets for ordinal and temporal data

Part V

Conclusion

Conclusions and Future Work

8.1. Conclusions

The World Wide Web has evolved and nowadays the Semantic Web has grown from a vision to a reality. The amount of semantic data available in the Web is rapidly increasing, especially thanks to Linked Data principles and best practices, which have been adopted by a growing number of data providers. Knowledge is available and shared at an unprecedented scale, but such abundance of data can be useless without effective ways for users to interact with it. The Semantic Web poses significant interaction challenges but also opportunities for users.

For lay-users it is difficult to explore and use this data with existing tools. Structured query languages such as SPARQL allow users to submit queries that can solve their information needs, but require them to be familiar with Semantic Web technologies and with the underlying data structure. In order for users to use and understand this data, we must explore effective ways for displaying, browsing and querying Semantic Web data.

La World Wide Web ha evolucionat i avui en dia la Web Semàntica ha passat de ser una visió a una realitat. La quantitat de dades semàntiques disponibles a la Web està augmentant ràpidament, en especial gràcies a iniciatives com Linked Data, que han estat adoptades per un gran nombre de proveïdors de dades. Les dades estan disponibles i compartides a gran escala, però tal quantitat de dades pot ser inútil si els usuaris no disposen d'eines adequades per explorar-les. La Web Semàntica planteja reptes respecte a la interacció amb les dades, però també noves oportunitats per als usuaris.

Per a molts usuaris és difícil explorar i utilitzar aquestes dades amb les eines existents. Llenguatges com SPARQL permeten als usuaris formular consultes que poden resoldre les seves necessitats i trobar la informació que busquen, però requereixen estar familiaritzat amb tecnologies de Web Semàntica i conèixer l'estructura de les dades. Per tal de que els usuaris puguin entendre i utilitzar aquestes dades, hem d'explorar maneres efectives de presentar-les, explorar-les i consultar-les.

After some rounds of development and testing with end-users, it is possible to conclude that the main hypotheses of this work were correctly posed. We have validated our work with formal user studies and we have compared it against existing tools with better results. *Rhizomer* is the result of an iterative design process aimed at supporting Semantic Web data exploration and visualization. Our evaluations have motivated the introduction of new features and components, which lead to substantial improvements in effectiveness, efficiency and in users' satisfaction. Moreover, compared to other user studies with similar tools, we have involved non-expert users during the whole development process. *Rhizomer* has been evaluated with lay-users as part of a User-Centered Design development process.

Rhizomer is capable of publishing semantic data while facilitating user awareness of what information is contained in the dataset. Awareness is accomplished by components borrowed from the Information Architecture discipline and generated automatically from the dataset structure and ontologies. These IA components are able to use the semantics captured by ontologies and semantic data, providing users different ways to access and interact with the data. These components, which are automatically generated, facilitate publishing and browsing a dataset without requiring a priori knowledge of it or experience in Semantic Web tools.

First of all, users can obtain an overview of the data thanks to different components. With navigation menus users can see the main kinds of items in the dataset. Site maps and treemaps are useful to get a general view of the overall dataset structure. And the site index provides access to particular content. These components are useful for obtaining a broad view of the datasets, their main types and the relationships between them. A good overview provides users an appreciation of the collection of objects and can help achieving their information seeking goals.

Després de varies iteracions de desenvolupament i proves amb usuaris, és possible concloure que les hipòtesis principals d'aquesta tesis estaven ben plantejades. Aquest treball s'ha validat mitjançant proves amb usuaris i s'ha comparat amb eines existents, obtenint millors resultats. Rhizomer és el resultat d'un disseny iteratiu destinat a l'exploració i visualització de dades semàntiques. Les avaluacions han motivat la introducció de noves funcionalitats i components que han dut a millores substancials en eficàcia, eficiència i satisfacció dels usuaris. A més a més, en comparació amb altres eines similars, hem involucrat a usuaris finals durant tot el procés de desenvolupament. Rhizomer ha estat avaluat amb usuaris no experts com a part d'un procés de disseny centrat en l'usuari.

Rhizomer és capaç de publicar dades semàntiques i facilitar als usuaris l'accés i poder entendre la informació que contenen aquests conjunts de dades. Això és possible gràcies a components adaptats de la disciplina de l'Arquitectura de la Informació, generats automàticament a partir de l'estructura i ontologies del conjunt de dades. Els components capturen i utilitzen la semàntica definida en les ontologies i dades, proporcionant als usuaris diferents formes d'accedir i interactuar amb elles. Aquests components es generen automàticament i faciliten la publicació i exploració de conjunts de dades semàntics sense necessitat de tenir experiència en eines i tecnologies de Web Semàntica.

Els usuaris poden obtenir una vista global del conjunt de dades a través de diferents components. Amb els menús de navegació poden veure els tipus de dades més comuns. Els mapes del lloc i treemap són útils per a tenir una vista general de l'estructura de les dades. I l'índex del lloc proporciona accés a continguts determinats. Aquests components permeten als usuaris fer-se una idea de les dades existents, els principals tipus i les relacions entre ells; ajudant-los a aconseguir als seus objectius.

8.1. CONCLUSIONS

Once users have overviewed the dataset and detected the types of entities they are interested in, it is time to explore them. Faceted navigation allow users to perform an exploratory search. They show the most significant properties for different kinds of items and their values. Facets are ranked according to their frequency but also considering their descriptive value. Facet widgets and pivoting enable users to build complex queries to explore the data, without requiring to learn complicated technologies or knowing the vocabularies used in the explored datasets. Breadcrumbs provide context information to guide users to their target, showing them their path taken so far and allowing them to go back to previous steps in the exploration process.

Finally, once users have selected those resources they are interested in, they can get more details about them, see their properties and values. Moreover, specific visualizations such as maps, timelines and charts are available when they are compatible with the selected resources. These visualizations have been implemented following the Linked Data Visualization Model (LDVM), connecting data with visualizations dynamically. The LDVM describes the process of creating visualizations from Linked Data in a generic way, offering developers guidance on how to create visualizations for RDF data. By applying this model, developers and designers can obtain a better understanding of the visualization process with data stages, transformations and operators.

It is important to notice that the quality of the automatically generated interface and components depends on the quality of the data. Components such as the sitemap or treemap are created from the class or topic hierarchies obtained from ontologies, which are not always included in datasets.

Un cop els usuaris han obtingut una visió general del conjunt de dades i detectat els principals tipus d'entitats que els interessin, és hora d'explorar-los. La navegació per facetes permet als usuaris realitzar una cerca exploratòria. Les facetes mostren les propietats més rellevants per diferents tipus de dades i els seus valors. Les facetes es mostren d'acord a un rànquing que té en compte la seva freqüència i també el seu valor descriptiu. Funcionalitats com el pivotat o "widgets" permeten als usuaris construir consultes complexes per explorar les dades, sense necessitat d'aprendre tecnologies complicades o conèixer els vocabularis utilitzants en els conjunts de dades. Les molles de pa (breadcrumbs) proporcionen informació contextual per a guiar els usuaris fins al seu objectiu, els permeten veure el camí que han seguit i retornar a pàgines visitades anteriorment.

Finalment, quan els usuaris han seleccionat aquells recursos d'interès, poden obtenir més detalls sobre ells, veure les seves propietats i valors. A més a més, les dades es poden veure en visualitzacions específiques com mapes, línies temporals o gràfiques, sempre i quan estiguin disponibles. Aquestes visualitzacions s'han implementat seguint el Linked Data Visualization Model (Model de Visualització de Dades Enllaçades), lligant dinàmicament les dades amb visualitzacions. El LDVM descriu el procés de crear visualitzacions de dades enllaçades d'una manera genèrica i pot servir de guia a l'hora de crear visualitzacions de dades en RDF. Seguint aquest model, els desenvolupadors i dissenyadors poden entendre el procés de visualització dividit en estats de dades, transformacions i operadors.

És important destacar que la qualitat de la interfície generada automàticament depèn de la qualitat de les dades. Alguns components com el mapa del lloc web o el treemap es creen a partir de la jerarquia de classes o temes que s'obtenen de les ontologies. Malauradament, no tots els conjunts de dades disposen d'una ontologia.

For functionalities like pivoting or facet widgets it is necessary to know properties range, which sometimes is not specified. Furthermore, it is common to see properties with a defined range whose values do not match that range, or properties with different URIs that contain the same information. These issues occur particularly in datasets extracted from collaborative websites, which is the case of the DBpedia or LinkedMDB. In such cases, when data is not well defined and normalized, the quality of the user experience decreases.

Overall, our findings can be used as guidelines for designers of tools and websites based on semantic data. They can decide which components and functionalities to implement depending on the concrete tasks their users need to solve and the characteristics of their datasets.

Altres funcionalitats com “widgets” o el pivotat en les facetes necessiten saber el rang de cada propietat, i aquests no sempre estan especificats. D'altra banda, és comú trobar propietats amb un rang definit però que els seus valors no concorden amb aquest rang o propietats amb diferents URIs que contenen la mateixa informació. Aquestes situacions es donen sovint en conjunts de dades extrets de llocs web col·laboratius, cas de la DBpedia o LinkedMDB, els conjunts de dades utilitzats en les nostres avaluacions amb usuaris. En aquests casos, quan les dades no estan ben definides i normalitzades, la qualitat de l'experiència d'usuari empitjora.

Globalment, els resultats obtinguts en aquesta tesi poden servir com a guia per a dissenyadors d'eines i llocs web basats en dades semàntiques. Els dissenyadors poden decidir quins components i funcionalitats implementar en el seu lloc web depenent de les tasques que hauran de realitzar els seus usuaris i de les característiques dels conjunts de dades.

Use cases and demonstrations

Rhizomer is an open source project and its code can be found at Google Code³³. The main strength of *Rhizomer* is that it is independent from the application domain. A basic set of components is generated automatically from the dataset structure, taking into account both ontologies and the actual triples in the dataset. This basic set of components allows users to explore the data through overviews, facets that provide filtering and pivoting, and visualizations.

These are the two main demos of *Rhizomer*:

- **DBpedia:** it is one of the most famous and big Linked Open Data sets. DBpedia data is extracted from Wikipedia and mapped to a domain ontology. It was used in our evaluation in iteration 3. The demo is available at <http://rhizomik.net/dbpedia>.
- **LinkedMDB:** this dataset is generated from the Internet Movie Database (IMDb)³⁴. Data is extracted from the IMDb site and represented as Linked Data in RDF. We used this dataset in our evaluations in iterations 1, 2 and 4. The demo is available at <http://rhizomik.net/linkedmdb>.

It is important to highlight that all these demos are generated automatically just by deploying *Rhizomer* on top of the corresponding dataset, served using Jena, Virtuoso or OWLIM. More demos as well as previous versions of *Rhizomer* are available in the project website³⁵.

8.2. Publications

The contributions presented in this work have led to the following peer-reviewed publications in international conferences, workshops and scientific journals. They are related to different aspects of this work.

First of all, the work in this thesis was presented and discussed in the Doctoral Consortium of the 13th Conference on Human-Computer Interaction (Interact 2011):

- Brunetti, J.M., García, R.: **Information Architecture Automatization for the Semantic Web**. *Human-Computer Interaction - INTERACT 2011 - 13th IFIP TC 13 International Conference*, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part IV. Springer 2011 Lecture Notes in Computer Science ISBN 978-3-642-23767-6

Based on the tasks for data analysis proposed by Shneiderman [Shn96], we identified interaction patterns for exploring Linked Data and proposed a set of components to implement them:

³³<https://code.google.com/p/rhizomer/>

³⁴<http://www.imdb.com/>

³⁵<http://rhizomik.net/rhizomer>

- Gil, R.; López-Muzás, A.; Brunetti, J.M.; Gimeno, J.M.; García, R.: **Evaluating Interaction Patterns for Linked Data**. *1st International Conference on Web Intelligence, Mining and Semantics, WIMS'11, Late-Breaking News session*. May 25-27, 2011, Sogndal, Norway.

Navigation menus, facets and breadcrumbs were the first components implemented in our first iteration. They were presented together with the first evaluation results in the following publications:

- García, R.; Brunetti, J.M.; Gimeno, J.M.; Gil, R.: **Componentes de Arquitectura de la Información basados en Tecnologías de la Web Semántica**. *Actas del XI Congreso Internacional de Interacción Persona Ordenador (Interacción 2010)*. Garceta Grupo Editorial, pp. 207-216, 2010. ISBN: 978-84-92812-52-3
- García, R.; Brunetti, J.M.; López-Muzás, A.; Gimeno, J.M.; Gil, R.: **Publishing and Interacting with Linked Data**. *1st International Conference on Web Intelligence, Mining and Semantics, WIMS'11*, May 25-27, 2011, Sogndal, Norway. International Conference Proceedings Series, ACM. ISBN 978-1-4503-0148-0.
- García, R.; Brunetti, J.M.; López-Muzás, A.; Gimeno, J.M.; Gil, R.: **Interacting with Linked Data through an Automatic Information Architecture**. *Tecnologías de Linked Data y sus aplicaciones en España, TLDE'11, Workshop at CAEPIA'11*. November 7-10, 2011, Tenerife, Spain.
- Brunetti, J.M.; Gil, R.; Gimeno, J.M.; García, R.: **Improved Linked Data Interaction through an Automatic Information Architecture**. *International Journal of Software Engineering and Knowledge Engineering* 22(3): 325-344(2012)
- Brunetti, J.M.; Gil, R.; López-Muzás, A.; Gimeno, J.M.; García, R.: **Evaluación de una plataforma semántica para la Interacción con la Web de Datos**. *XII Congreso Internacional de Interacción Persona Ordenador, Interacción'11*. September 2-5, 2011, Lisboa, Portugal.

In our second iteration, we implemented pivoting and literal breadcrumbs. These functionalities and our second evaluation with end-users were presented in the following publications:

- Brunetti, J.M.; Gil, R.; García, R.: **Facets and Pivoting for Flexible and Usable Linked Data Exploration**. *Interacting with Linked Data Workshop, ILD'12, 9th Extended Semantic Web Conference, ESWC'12*. May 28, 2012, Crete, Greece. CEUR Workshop Proceedings, Vol. 913, pp. 22-35, 2012.
- Brunetti, J.M.; García, R.; Auer, S.: **From Overview to Facets and Pivoting for Interactive Exploration of Semantic Web Data**. *International Journal of Semantic Web and Information Systems*, Vol. 9, No. 1, pp. 1-20. IGI Global, 2013. ISSN 1552-6283.

We proposed SWET-QUM to evaluate the quality in use of Semantic Web exploration tools. The proposed model and the evaluation of *Rhizomer*, *SParallax* and *Virtuoso Faceted Browser* were presented in:

8.2. PUBLICATIONS

- González, J.L.; García, R.; Brunetti, J.M.; Gil, R.; Gimeno, J.M.: **SWET-QUM: a quality in use extension model for Semantic Web exploration tools.** (Honorable Mention). *13th International Conference on Interacción Persona-Ordenador, INTERACCION '12.* October 3-5, 2012, Elche, Spain. International Conference Proceedings Series (ICPS), ACM, pp. 15:1-15:8. ISBN 978-1-4503-1314-8.
- González, J.L.; García, R.; Brunetti, J.M.; Gil, R.; Gimeno, J.M.: **Using SWET-QUM to Compare the Quality in Use of Semantic Web Exploration Tools.** *Journal of Universal Computer Science, Vol. 19, No. 8, pp. 1025-1046. J.UCS, 2013.* ISSN 0948-6968.

In our third iteration, we proposed different components to provide an overview of Semantic Web datasets. These components and their evaluation were presented in:

- Brunetti, J.M.: **Design and evaluation of overview components for effective semantic data exploration.** *In Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics (WIMS '13).* Madrid, Spain.

We proposed the Linked Data Visualization Model, which was presented in the Posters & Demos session in the International Semantic Web Conference ISWC'12. This work has been extended, resulting in another publication in reviewing process:

- Brunetti, J.M.; Auer, S.; García, R.: **The Linked Data Visualization Model.** *11th International Semantic Web Conference, ISWC'12, Posters & Demonstrations Track.* November 11-15, 2012, Boston, MA, USA. CEUR Workshop Proceedings, Vol. 914, pp. 5-8, 2012.
- Brunetti, J.M.; Auer, S.; García, R. Klímek J., Nežaský M.: **Formal Linked Data Visualization Model.** *The 15th International Conference on Information Integration and Web-based Applications & Services, IIWAS '13.* 2-4 December 2013, Vienna, Austria.

Finally, *Rhizomer* was presented in the IESD'13 Workshop at Hypertext'13, being the winner of the IESD Challenge:

- García, R.; Brunetti, J.M.; Gil, R.; Gimeno, J.M.: **Rhizomer: Overview, Facets and Pivoting for Semantic Data Exploration.** *Intelligent Exploration of Semantic Data (IESD'13).* Workshop at Hypertext'13. May 1, 2013. Paris, France. Winner of the IESD Challenge.

8.3. Future Work

While the growth of Semantic Web data generates new problems and challenges, it also offers new chances and opens many possibilities for future work. The future work can be classified in different lines.

Further iterations and components

Our future work focuses on improving the Information Architecture components and their usability. First of all, the proposals in iteration 5 must be implemented and it is also necessary to solve the remaining minor issues identified in other iterations. Once fixed these issues, a new user evaluation should be performed, evaluating these new functionalities.

Usability evaluation needs to be extended, considering other user profiles and techniques such as heuristic evaluation [NM90], in order to obtain new feedback about the user interface and identify other usability problems. We will also further study these components with other datasets to evaluate how they adapt to different scenarios depending on their size and structure, and also how users perform similar tasks using them.

Improve overviews

The overview components we propose are based on the dataset schema and ontologies. This is a limitation for those datasets without a schema. A possibility to create overviews for those datasets without schema is to apply clustering algorithms. Clustering is the process of classifying similar resources into groups called clusters. This could allow us to create navigation menus with a limited number of options that represent the whole dataset.

We should also experiment more with our algorithm to generate navigation menus. As a result of executing the algorithm, the class hierarchy is modified and some classes may disappear, while others can be grouped in new classes. Altering the hierarchy can be disorienting for some users and they may lose context information.

Extending faceted browsing

Preliminary work has been done to improve facet ranking and faceted navigation for RDF data. We have proposed a new method for ranking facets according to their descriptive value, but further research and experimentation is needed in order to develop more heuristics that can find intuitive facets for users.

Regarding the user interface, the objective is to extend facet widgets for other kind of values being managed, i.e. alphabetical values, dates, geographical points, hierarchical values, etc. A variety of widgets for faceted browsing is presented in [Pol09]. Since the number of different values for a facet can be overwhelming, clustering should be also considered.

8.3. FUTURE WORK

Another objective is to improve the functionality of this component by adding other operators for restricting the results: inverse selection, existential selection, join selection, selection between ranges, etc. It would be useful to allow queries that can represent other operations, for instance conjunctions of values within one facet, e.g. “Actors starring in Star Wars Episode 1 and also in Star Wars Episode II”. The challenge is not only to incorporate such extended functionalities, but also to keep the user interface easily usable.

Details-on-demand and visualizations

The Linked Data Visualization Model is the first step on a larger research agenda aiming at automatizing the visualization of semantic data. In future work we focus on complementing the Linked Data Visualization Model with an ontology that can help during the matching process between data and visualizations, capture the intermediate data structures that can be generated and choose the visualizations more suitable for each data structure. We also plan to extend the library of visualizations, which will facilitate the creation of an ecosystem of data publication and data visualization approaches, which can co-exist and evolve independently.

Performance and scalability issues

The evaluation with users showed the system’s scalability with large datasets such as DBPedia or LinkedMDB. While the user interface allows users to interact with data with acceptable response times, some operations with large datasets have a negative impact on the UI because it takes some time to compute SPARQL queries. Consequently, the process of generating the User Interface components needs to be precomputed.

One approach to increase the performance is to reduce the requirements of the interface and turn off some features. For example, the SPARQL queries to obtain the cardinality of facet values or to detect the range of each facet when it is not defined are expensive to calculate. By removing such features the performance can be improved. Alternatively, we can explore approaches to optimize SPARQL queries [MBV12, LN13]. These issues should be addressed for future versions.

SWET-QUM

Future work is also aimed at improving SWET-QUM and extending it to other kinds of tools based on Semantic Web technologies. Currently, the priority is to explore additional metrics that can enrich the model. For instance, a metric related to the number of interaction steps that users needed to complete a task could be related with Layout Flexibility.

Other metrics under consideration are those that exploit techniques like Eye Tracking, which help understanding user interaction better, or that analyse how the interface adapts to user characteristics (adaptability) or preferences (personalisation). The proposal is to revise the UI Component Efficiency metric and base it on the total time users spent looking at the interface, including time not looking at any UI component under consideration. Right now, only the time looking at these UI components is considered.

APPENDIX A

User Evaluation Documents

Consentimiento de participación y grabación

En cumplimiento de la Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal (LOPD), mediante el presente documento confirmo que:

1. Acepto participar en esta prueba de usuario que lleva a cabo el laboratorio de usabilidad UsabiliLAB.
2. Autorizo la filmación en vídeo de la prueba.
3. Esta grabación podrá ser utilizada con finalidades científicas para el análisis de los datos recogidos en el proyecto o para divulgar los resultados, ya sea por parte del GRIHO o por la empresa cliente en presentaciones o reuniones profesionales.
4. Renuncio a los derechos de la grabación de vídeo y entiendo que la grabación se puede utilizar para los fines descritos sin permiso adicional.
5. En ningún caso se podrá hacer un uso que pueda vulnerar mi imagen o dignidad personal ni hacer un uso comercial.
6. Puedo ejercitar los derechos de acceso, rectificación, cancelación y oposición de mis datos personales, de acuerdo a la normativa vigente, comunicándolo a los datos de contacto de los que dispongo.
7. He tomado esta decisión basándome en la información que se me ha proporcionado por escrito y he tenido la oportunidad de recibir información adicional en caso de haberla solicitado.
8. Entiendo que la participación es voluntaria y que puedo retirar este consentimiento en cualquier momento sin recibir una penalización por ello.

Nombre y apellidos del participante:

DNI: _____ Teléfono: _____ Email: _____

Fecha: _____

Firma participante

Firma administrador del test

Para más información o para cualquier tema relacionado con el proyecto se puede usted dirigir a:

Josep Maria Brunetti
Universitat de Lleida
josepmbunetti@diei.udl.cat

Universidad de Lleida



ID participante: _____

Cuestionario Post-Test

Fecha: _____

| Preguntas | De acuerdo ----- En desacuerdo | | | | |
|---|--------------------------------|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 |
| Ha sido fácil utilizar esta herramienta. | | | | | |
| El sistema es intuitivo. | | | | | |
| Me he divertido utilizándolo. | | | | | |
| Las opciones disponibles son fácilmente identificables. | | | | | |

1. ¿Qué es lo que más te ha gustado/disgustado del sistema? ¿Por qué?

2. ¿Cómo te ha parecido la interacción en general con el sistema?

3. ¿Qué cosas cambiarías o mejorarías en esta herramienta?

4. ¿Tienes alguna sugerencia adicional que quisieras compartir?

- [ABDM04] Riccardo Albertoni, Alessio Bertone, and Monica De Martino. Semantic Web and Information Visualization. In *Semantic Web Applications and Perspectives (SWAP)1st Italian Semantic Web Workshop*, 2004.
- [ABK⁺07] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *In 6th Intl Semantic Web Conference, Busan, Korea*, pages 11–15. Springer, 2007.
- [ADR06] Sören Auer, Sebastian Dietzold, and Thomas Riechert. Ontowiki – a tool for social, semantic collaboration. In *Proceedings of the 5th international conference on The Semantic Web, ISWC'06*, pages 736–749, Berlin, Heidelberg, 2006. Springer-Verlag.
- [AH09] Keith Alexander and Michael Hausenblas. Describing linked datasets - on the design and usage of void, the vocabulary of interlinked datasets. In *In Linked Data on the Web Workshop (LDOW 09), in conjunction with 18th International World Wide Web Conference (WWW 09)*, 2009.
- [AWS92] Christopher Ahlberg, Christopher Williamson, and Ben Shneiderman. Dynamic queries for information exploration: an implementation and evaluation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '92*, pages 619–626, New York, NY, USA, 1992. ACM.
- [BAI05] James Blustein, Ishtiaq Ahmed, and Keith Instone. An evaluation of look-ahead breadcrumbs for the www. In *Proceedings of the sixteenth ACM conference on Hypertext and hypermedia, HYPERTEXT '05*, pages 202–204, New York, NY, USA, 2005. ACM.
- [Bat89] Marcia J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Information Review*, 13(5):407–424, 1989.
- [BCH07] Chris Bizer, Richard Cyganiak, and Tom Heath. How to publish Linked Data on the Web, 2007.
- [Bec04] D. Beckett. Rdf/xml syntax specification (revised). Technical report, W3C Recommendation, 2004.
- [Ber88] Mark Bernstein. The bookmark and the compass: orientation tools for hypertext users. *SIGOIS Bull.*, 9(4):34–45, October 1988.
- [Bev01] N. Bevan. International standards for HCI and usability. *International Journal of Human-Computer Studies*, 55(4):533–552, October 2001.

BIBLIOGRAPHY

- [BH11] Sören Brunk and Philipp Heim. tfacet: Hierarchical faceted exploration of semantic data using well-known interaction concepts. In *Proceedings of the International Workshop on Data-Centric Interactions on the Web (DCI 2011)*, volume 817 of *CEUR-WS.org*, pages 31–36, 2011.
- [BHBL09] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22, MarMar 2009.
- [BHvW00] M. Bruls, K. Huizing, and J. van Wijk. Squarified Treemaps. In *Proc. of Joint Eurographics and IEEE TCVG Symp. on Visualization (TCVG 2000)*, pages 33–42. IEEE Press, 2000.
- [BL06] Tim Berners-Lee. Linked data - design issues. *W3C*, (09/20), 2006.
- [BICC⁺06] Tim Berners-lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.
- [BICP92] Tim Berners-lee, Robert Cailliau, and Bernd Pollermann. World-wide web: The information universe. *Communications of the ACM*, 37:76–82, 1992.
- [BLF99] Tim Berners-Lee and Mark Fischetti. *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper San Francisco, September 1999.
- [BIHL⁺] T Berners-lee, J. Hollenbach, Kanghao Lu, J. Presbrey, and Mc Schraefel. Tabulator redux: Browsing and writing linked data.
- [BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.
- [BMV11] Roi Blanco, Peter Mika, and Sebastiano Vigna. Effective and efficient entity search in rdf data. In *Proceedings of the 10th international conference on The semantic web - Volume Part I, ISWC'11*, pages 83–97, Berlin, Heidelberg, 2011. Springer-Verlag.
- [BN01] S. Todd Barlow and Padraic Neville. A comparison of 2-d visualizations of hierarchies. In *INFOVIS*, pages 131–138, 2001.
- [BOH11] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.
- [BPGB07] Uldis Bojars, Alexandre Passant, Frederick Giasson, and John Breslin. An architecture to discover and query decentralized RDF data. In Sören Auer, Chris Bizer, Tom Heath, and Gunnar AAstrand Grimnes, editors, *Proceedings of 3rd ESWC Workshop on Scripting for the Semantic Web (SFSW07)*, volume 248 of *CEUR Workshop Proceedings ISSN 1613-0073*, June 2007.
- [Bro95] Frederick P. Brooks. *The mythical man-month : essays on software engineering*. Addison-Wesley Pub. Co, anniversary edition, August 1995.

BIBLIOGRAPHY

- [BW84] Victor R. Basili and David Weiss. A methodology for collecting valid software engineering data. *IEEE Computer Society Trans. Software Engineering*, 10(6):728–738, 1984.
- [BW06] Holger Bast and Ingmar Weber. When you're lost for words: Faceted search with autocompletion. In Andrei Broder and Yoelle Maarek, editors, *SIGIR'06 Workshop on Faceted Search*, pages 31–35, Seattle, USA, August 2006. ACM.
- [CCPD09] Stephane Corlosquet, Richard Cyganiak, Axel Polleres, and Stefan Decker. RDFa in Drupal: Bringing cheese to the web of data. In Sören Auer, Chris Bizer, and Gunnar AAstrand Grimnes, editors, *Proc. of 5th Workshop on Scripting and Development for the Semantic Web at ESWC 2009*, volume 449 of *CEUR Workshop Proceedings ISSN 1613-0073*, June 2009.
- [CCS93] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User-Analysis: An IT Mandate, 1993.
- [CDD⁺04] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, WWW Alt. '04*, pages 74–83, New York, NY, USA, 2004. ACM.
- [Chi00] Ed H. Chi. A taxonomy of visualization techniques using the data state reference model. In *Proceedings of the IEEE Symposium on Information Vizualization 2000, INFOVIS '00*, pages 69–, Washington, DC, USA, 2000. IEEE Computer Society.
- [cif04] Common industry format for quality in use test reports (annex f); software engineering - product quality - part 4: Quality in use metrics, iso/iec 9126-4. Technical report, International Organization for Standardization, 2004.
- [CLCGP06] Óscar Corcho, Angel López-Cima, and Asunción Gómez-Pérez. The odesew 2.0 semantic web application framework. In Les Carr, David De Roure, Arun Iyengar, Carole A. Goble, and Michael Dahlin, editors, *WWW*, pages 1049–1050. ACM, 2006.
- [CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
- [CMS99] Stuart K. Card, J. D. Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Academic Press, London, 1999.
- [CNF09] Edward C. Clarkson, Shamkant B. Navathe, and James D. Foley. Generalized formal models for faceted user interfaces. In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries, JCDL '09*, pages 125–134, New York, NY, USA, 2009. ACM.
- [CPJ05] Dave Crane, Eric Pascarello, and Darren James. *Ajax in Action*. Manning Publications, October 2005.
- [CS96] P. Chandler and J. Sweller. Cognitive load while learning to use a computer program. *Applied Cognitive Psychology.*, 10:151–170, 1996.

BIBLIOGRAPHY

- [dASB09] S. F. C. de Araújo, D. Schwabe, and S. D. J. Barbosa. Experimenting with explorator: a direct manipulation generic rdf browser and querying tool. In *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009)*, February 2009.
- [DFAB97] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-computer interaction*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.
- [DHDZ10] Stephen Davies, Jesse Hatfield, Chris Donaher, and Jessica Zeitz. User interface design considerations for linked data authoring environments. In Christian Bizer, Tom Heath, Tim Berners-Lee, and Michael Hausenblas, editors, *LDOW*, volume 628 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.
- [DIW05] Wisam Dakka, Panagiotis G. Ipeirotis, and Kenneth R. Wood. Automatic construction of multifaceted browsing interfaces. In *Proceedings of the 14th ACM international conference on Information and knowledge management, CIKM '05*, pages 768–775, New York, NY, USA, 2005. ACM.
- [DKS07] Leonidas Deligiannidis, Krys J. Kochut, and Amit P. Sheth. Rdf data exploration and visualization. In *Proceedings of the ACM first workshop on CyberInfrastructure: information management in eScience, CIMS '07*, pages 39–46, New York, NY, USA, 2007. ACM.
- [DR11] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.
- [DRM⁺08] Debabrata Dash, Jun Rao, Nimrod Megiddo, Anastasia Ailamaki, and Guy Lohman. Dynamic faceted search for discovery-driven analysis. In *Proceedings of the 17th ACM conference on Information and knowledge management, CIKM '08*, pages 3–12, New York, NY, USA, 2008. ACM.
- [DRP11] Aba-Sah Dadzie, Matthew Rowe, and Daniela Petrelli. Hide the stack: toward usable linked data. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Volume Part I, ESWC'11*, pages 93–107, Berlin, Heidelberg, 2011. Springer-Verlag.
- [EF10] Niklas Elmqvist and Jean-Daniel Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Trans. Vis. Comput. Graph.*, 16(3):439–454, 2010.
- [EH88] Deborah M. Edwards and Lynda Hardman. 'lost in hyperspace': Cognitive mapping and navigation in a hypertext environment. In *UK Hypertext*, pages 105–125, 1988.
- [EHS⁺02] Jennifer English, Marti Hearst, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Flexible search and navigation using faceted metadata. Technical report, University of Berkeley, 2002.
- [EII01] Ame Elliott. Flamenco image browser: using metadata to improve image search during architectural design. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems, CHI EA '01*, pages 69–70, New York, NY, USA, 2001. ACM.

BIBLIOGRAPHY

- [Erl] Orri Erling. Faceted views over large-scale linked data.
- [EVS11] Basil Ell, Denny Vrandečić, and Elena Simperl. Labels in the web of data. In *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ISWC'11, pages 162–176, Berlin, Heidelberg, 2011. Springer-Verlag.
- [Fer08] Sebastien Ferre. Agile browsing of a document collection with dynamic taxonomies. *Database and Expert Systems Applications, International Workshop on*, 0:377–381, 2008.
- [Gar10] Jesse James Garrett. *The Elements of User Experience: User-Centered Design for the Web and Beyond*. New Riders, Indianapolis, IN, United States, 2 edition, 2010.
- [GB04] Ramanathan V. Guha and Dan Brickley. RDF Vocabulary Description Language 1.0: RDF Schema. W3C recommendation, W3C, February 2004.
- [GC02] Vladimir Geroimenko and Chaomei Chen, editors. *Visualizing the Semantic Web*. Springer, 2002.
- [GGP⁺08] Roberto García, Juan Manuel Gimeno, Ferran Perdrix, Rosa Gil, and Marta Oliva. A platform for object-action semantic web interaction. In *Proceedings of the 16th international conference on Knowledge Engineering: Practice and Patterns*, EKAW '08, pages 404–418, Berlin, Heidelberg, 2008. Springer-Verlag.
- [GGP⁺10] Roberto García, Juan Manuel Gimeno, Ferran Perdrix, Rosa Gil, Marta Oliva, Juan Miguel López, Afra Pascual, and Montserrat Sendín. Building a usable and accessible semantic web interaction platform. *World Wide Web*, 13(1-2):143–167, March 2010.
- [GMPS00] Stephan Greene, Gary Marchionini, Catherine Plaisant, and Ben Shneiderman. Previews and overviews in digital libraries: Designing surrogates to support visual information seeking. *Journal of the American Society for Information Science*, 51:380–393, 2000.
- [Gra03] Toni Granollers. User centred design process model, integration of usability engineering and software engineering. In *Proceedings of INTERACT 2003*, pages 673–675, 2003.
- [Gro08] W3C XML Working Group. The xml 1.0 standard (5th edition), 2008.
- [Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.
- [Har10] Andreas Harth. Visinav: A system for visual search and navigation on web data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4), 2010.
- [HBO] Jeffrey Heer, Michael Bostock, and Vadim Ogievetsky. A tour through the visualization zoo. *Queue*, 8(5):20:20–20:30.
- [HBS⁺10] Rasmus Hahn, Christian Bizer, Christopher Sahnwaldt, Christian Herta, Scott Robinson, Michaela Bürgle, Holger Düwiger, and Ulrich Scheel. Faceted wikipedia search. In *BIS*, pages 1–11, 2010.

BIBLIOGRAPHY

- [HCB08] Tuukka Hastrup, Richard Cyganiak, and Uldis Bojars. Browsing linked data with fenfire, 2008.
- [HDS06] Tom Heath, John Domingue, and Paul Shabajee. User interaction and uptake challenges to successfully deploying semantic web technologies. In *in Proc. 3rd International Semantic Web User Interaction Workshop*, 2006.
- [Hea00] Marti A. Hearst. Next generation web search: Setting our sites. *IEEE DATA ENGINEERING BULLETIN*, 23, 2000.
- [Hea06a] M. Hearst. Design recommendations for hierarchical faceted search interfaces. *ACM SIGIR Workshop on Faceted Search*, 2006.
- [Hea06b] Marti A. Hearst. Clustering versus faceted categories for information exploration. *Commun. ACM*, 49(4):59–61, apr 2006.
- [Hea08a] Marti A. Hearst. Uis for faceted navigation: Recent advances and remaining open problems. In *in the Workshop on Computer Interaction and Information Retrieval, HCIR 2008*, 2008.
- [Hea08b] Tom Heath. How will we interact with the web of data? *IEEE Internet Computing*, 12(5):88–91, September 2008.
- [HEE⁺02] Marti Hearst, Ame Elliott, Jennifer English, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Finding the flow in web site search. *Commun. ACM*, 45(9):42–49, September 2002.
- [Hen07] Monika Henzinger. Search Technologies for the Internet. *Science*, 317(5837):468–471, July 2007.
- [HHG09] Christian Hirsch, John Hosking, and John Grundy. Interactive visualization tools for exploring the semantic graph of large knowledge spaces. In *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009)*, February 2009.
- [HK09] David Huynh and David Karger. Parallax and Companion: Set-based Browsing for the Data Web. 2009.
- [HKM07] David F. Huynh, David R. Karger, and Robert C. Miller. Exhibit: lightweight structured data publishing. In *Proceedings of the 16th international conference on World Wide Web, WWW '07*, pages 737–746, New York, NY, USA, 2007. ACM.
- [HL90] I. Hamilton and A. Life. *Simulation And The User Interface*. Taylor & Francis, 1990.
- [HMM00] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, January 2000.
- [HMS⁺05] Eero Hyvönen, Eetu Mäkelä, Mirva Salminen, Arttu Valo, Kim Viljanen, Sampsa Saarela, Miikka Junnila, and Suvi Kettula. Museumfinland-finnish museums on the semantic web. *Web Semant.*, 3(2-3):224–241, October 2005.

BIBLIOGRAPHY

- [HR07] Jeffrey Heer and George Robertson. Animated transitions in statistical data graphics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1240–1247, November 2007.
- [HRH08] W. Halb, Y. Raimond, and M. Hausenblas. Building Linked Data For Both Humans and Machines. In *WWW 2008 Workshop: Linked Data on the Web (LDOW2008)*, Beijing, China, 2008.
- [HSM⁺10] Wolfgang Halb, Alexander Stocker, Harald Mayer, Helmut Mülner, and Ilir Ademi. Towards a commercial adoption of linked open data for online content providers. In *Proceedings of the 6th International Conference on Semantic Systems, I-SEMANTICS '10*, pages 16:1–16:8, New York, NY, USA, 2010. ACM.
- [HSTS10] Olaf Hartig, Juan Sequeda, Jamie Taylor, and Patrick Sinclair. How to consume linked data on the web: tutorial description. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *WWW*, pages 1347–1348. ACM, 2010.
- [HvOH06] Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. /facet: A Browser for Heterogeneous Semantic Web Repositories. In *ISWC*, 2006.
- [HZ08] Philipp Heim and Jürgen Ziegler. Handling the complexity of rdf data: Combining list and graph visualization. In *Proceedings of the 8th International Conference on Knowledge Management (I-KNOW 08)*, pages 324–331. Graz: J.UCS, 2008.
- [HZL08] Philipp Heim, Jürgen Ziegler, and Steffen Lohmann. gFacet: A browser for the web of data. In *Proceedings of the International Workshop on Interacting with Multimedia Content in the Social Semantic Web (IMC-SSW 2008)*, volume 417 of *CEUR-WS*, pages 49–58, 2008.
- [IfS03] International Organization for Standardization ISO and International Organization for Standardization. *Ergonomics of Human-system Interaction: Guidance on Accessibility for Human-computer Interfaces*. International standard. ISO, 2003.
- [Ins] K Instone. Location, path and attribute breadcrumbs. In *3rd Annual Information Architecture Summit*.
- [Int99] International Standards Organization. *ISO 13407. Human Centred Design Process for Interactive Systems*. Geneva, Swiss, 1999.
- [Int00] International Organization for Standardization. *ISO/IEC 9000:2000 Quality Management Systems – Fundamentals and vocabulary*. Technical report, International Organization for Standardization, 2000.
- [Int11] International Organization for Standardization. *Iso 25010-3, software product quality requirements and evaluation (square): Software product quality and system quality in use models*. Technical report, International Organization for Standardization, 2011.

BIBLIOGRAPHY

- [ISO98] ISO. ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability. Technical report, International Organization for Standardization, 1998.
- [iso01] Software engineering - product quality, ISO/IEC 9126-1. Technical report, International Organization for Standardization, 2001.
- [KD08] Georgi Kobilarov and Ian Dickinson. Humboldt: Exploring Linked Data. In *Linked Data on the Web Workshop (LDOW2008) at WWW2008*, Beijing, China, 2008.
- [KGJ⁺10] Oleg V. Komogortsev, Denise V. Gobert, Sampath Jayarathna, Do Hyong Koh, and Sandeep A. Munikrishne Gowda. Standardization of automated analyses of oculomotor fixation and saccadic behaviors. *IEEE Trans. Biomed. Engineering*, 57(11):2635–2645, 2010.
- [KHL⁺07] Akrivi Katifori, Constantin Halatsis, George Lepouras, Costas Vassilakis, and Eugenia Giannopoulou. Ontology visualization methods - a survey. *ACM Comput. Surv.*, 39(4), November 2007.
- [Kob04] Alfred Kobsa. User experiments with tree visualization systems. In *Proceedings of the IEEE Symposium on Information Visualization, INFOVIS '04*, pages 9–16, Washington, DC, USA, 2004. IEEE Computer Society.
- [KS06] David Karger and MC Schraefel. The pathetic fallacy of RDF. Position Paper for SWUI06, 2006.
- [KVV06] Markus Krötzsch, Denny Vrandečić, and Max Völkel. Semantic MediaWiki. In Isabel Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Mike Uschold, and Lora Aroyo, editors, *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, chapter 68, pages 935–942. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006.
- [Lew82] C. H. Lewis. Using the "Thinking Aloud" Method In Cognitive Interface Design. Technical Report RC-9265, IBM, 1982.
- [LN13] Johannes Lorey and Felix Naumann. Detecting sparql query templates for data prefetching. In *Proceedings of the 10th Extended Semantic Web Conference (ESWC)*, Montpellier, France, 0 2013.
- [lon] Simile: Longwell rdf browser. <http://simile.mit.edu/wiki/Longwell>.
- [mar] Marbles. <http://marbles.sourceforge.net/>.
- [Mar06] Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, April 2006.
- [MBV12] Sara Magliacane, Alessandro Bozzon, and Emanuele Della Valle. Efficient execution of top-k sparql queries. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7649 of *Lecture Notes in Computer Science*, pages 344–360. Springer, 2012.

BIBLIOGRAPHY

- [MF95] Sougata Mukherjea and James D. Foley. Visualizing the world-wide web with the navigational view builder. *Comput. Netw. ISDN Syst.*, 27(6):1075–1087, April 1995.
- [MG03] Paul Mutton and Jennifer Golbeck. Visualization of Semantic Metadata and Ontologies. In *IEEE Int. Conf. on Information Visualization*, London, 2003. IEEE.
- [Mil56] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63(2):81–97, March 1956.
- [MMP⁺08] Van Kleek M., Bernstein M., André P., Perttunen M., Karger D, and Schraefel MC. Simplifying knowledge creation and access for end users on the sw. SWUI Workshop at CHI 2008, April 5, Florence, Italy, 2008.
- [MPA07] Alistair Miles and José R. Pérez-Agüera. Skos: Simple knowledge organisation for the web. *Cataloging & Classification Quarterly*, 43(3):69–83, 2007.
- [msWRS06] m.c. schraefel, Max L. Wilson, Alistair Russell, and Daniel Alexander Smith. mspace: improving information access to multimedia domains with multimodal exploratory search. *Communication of the ACM*, 49(4):47–49, April 2006.
- [Muz09] Antonio López Muzás. Sistema modular de presentación de información para la plataforma de web semántica rhizomer. Master’s thesis, Universitat de Lleida, 09 2009.
- [MWT⁺02] Michael C. Medlock, Dennis Wixon, Mark Terrano, Ramon L. Romero, and Bill Fulton. Using the RITE method to improve products; a definition and a case study. In *Usability Professionals Association*, 2002.
- [Nie93] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.
- [Nie95] Jakob Nielsen. Usability inspection methods. In *Conference companion on Human factors in computing systems*, CHI '95, pages 377–378, New York, NY, USA, 1995. ACM.
- [NL94] Jakob Nielsen and Jonathan Levy. Measuring usability: preference vs. performance. *Commun. ACM*, 37(4):66–75, April 1994.
- [NL06] Jakob Nielsen and Hoa Loranger. *Prioritizing Web Usability*. New Riders, Berkeley, CA, 2006.
- [NM90] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, CHI '90, pages 249–256, New York, NY, USA, 1990. ACM.
- [Nor90] D.A. Norman. *The design of everyday things*. New York: Doubleday, 1990.
- [ODD06] Eyal Oren, Renaud Delbru, and Stefan Decker. Extending faceted navigation for rdf data. In *International Semantic Web Conference*, pages 559–572, 2006.

BIBLIOGRAPHY

- [OHS09] Antti Oulasvirta, Janne P. Hukkinen, and Barry Schwartz. When more is less: the paradox of choice in search engine use. In James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel, editors, *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*, pages 516–523. ACM, 2009.
- [PBKL06] Emmanuel Pietriga, Christian Bizer, David Karger, and Ryan Lee. Fresnel - a browser-independent presentation vocabulary for rdf. In *In: Proceedings of the Second International Workshop on Interaction Design and the Semantic Web*, pages 158–171. Springer, 2006.
- [PGB02] Catherine Plaisant, Jesse Grosjean, and Benjamin B. Bederson. Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, pages 57–, Washington, DC, USA, 2002. IEEE Computer Society.
- [PH99] Wanda Pratt and Marti A. Hearst. A knowledge-based approach to organizing retrieved documents. pages 80–85, 1999.
- [Pie06] Emmanuel Pietriga. Semantic web data visualization with graph style sheets. In *Proceedings of the 2006 ACM symposium on Software visualization, SoftVis '06*, pages 177–178, New York, NY, USA, 2006. ACM.
- [PK00] Joonah Park and Jinwoo Kim. Effects of contextual navigation aids on browsing diverse web systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '00*, pages 257–264, New York, NY, USA, 2000. ACM.
- [Pol09] Jan Polowinski. Widgets for faceted browsing. In MichaelJ. Smith and Gavriel Salvendy, editors, *Human Interface and the Management of Information. Designing Information Environments*, volume 5617 of *Lecture Notes in Computer Science*, pages 601–610. Springer Berlin Heidelberg, 2009.
- [Pre05] Roger S. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Higher Education, 6th edition, 2005.
- [PS06] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. Technical report, W3C, 2006.
- [PSDB99] Catherine Plaisant, Ben Shneiderman, Khoa Doan, and Tom Bruns. Interface and data architecture for query preview in networked information systems. *ACM Trans. Inf. Syst.*, 17(3):320–341, July 1999.
- [PSHS11] Igor O. Popov, M. C. Schraefel, Wendy Hall, and Nigel Shadbolt. Connecting the dots: a multi-pivot approach to data exploration. In *Proceedings of the 10th international conference on The semantic web - Volume Part I, ISWC'11*, pages 553–568, Berlin, Heidelberg, 2011. Springer-Verlag.
- [QK04] D. A. Quan and R. Karger. How to make a semantic web browser. In *Proceedings of the 13th international conference on World Wide Web, WWW '04*, pages 255–265, New York, NY, USA, 2004. ACM.

BIBLIOGRAPHY

- [Ran62] S. R. Ranganathan. *Elements of library classification*. Ranganathan Series in Library Science ; 8. Bombay : Asia Publishing House, third edition edition, 1962.
- [RC02] Mary Beth Rosson and John M. Carroll. *Usability engineering: scenario-based development of human-computer interaction*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.
- [Ret94] Marc Rettig. Prototyping for tiny fingers. *Commun. ACM*, 37(4):21–27, April 1994.
- [rgr] Rdf gravity. <http://semweb.salzburgresearch.at/apps/rdg-gravity/index.html>.
- [RM02] Louis Rosenfeld and Peter Morville. *Information Architecture for the World Wide Web*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2nd edition, 2002.
- [RMC91] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone Trees: animated 3D visualizations of hierarchical information. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, CHI '91, pages 189–194, New York, NY, USA, 1991. ACM.
- [ROH05] Lloyd Rutledge, Jacco Van Ossenbruggen, and Lynda Hardman. Making rdf presentable: Integrated global and local semantic web browsing, 2005.
- [RR07] Leonard Richardson and Sam Ruby. *Restful web services*. O'Reilly, first edition, 2007.
- [SAN13] Saeedeh Shekarpour, Sören Auer, and Axel-Cyrille Ngonga Ngomo. Question answering on interlinked data. In *Proceedings of WWW*, 2013.
- [SBLH06] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, May 2006.
- [SD04] G. Schreiber and M. Dean. OWL web ontology language reference. <http://www.w3.org/TR/2004/REC-owl-ref-20040210/>, February 2004.
- [SF09] G. M. Sacco and S. Ferré. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, volume 25 of *The Information Retrieval Series*, chapter 9 - Applications and Experiences, pages 263–302. Springer, 2009.
- [SG09] D. Spencer and J.J. Garrett. *Card Sorting: Designing Usable Categories*. Rosenfeld Media, LLC, 2009.
- [Shn92] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, January 1992.
- [Shn96] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Visual Languages*, number UMCP-CSD CS-TR-3665, pages 336–343, College Park, Maryland 20742, U.S.A., 1996.
- [SK98] Ian Sommerville and Gerald Kotonya. *Requirements Engineering: Processes and Techniques*. John Wiley & Sons, Inc., New York, NY, USA, 1998.

BIBLIOGRAPHY

- [SM10] Zhenning Shangguan and Deborah L. McGuinness. Towards faceted browsing over linked data. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, 2010.
- [Smi96] Pauline A. Smith. Towards a practical measure of hypertext usability. *Interact. Comput.*, 8(4):365–381, December 1996.
- [SP04] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition)*. Addison Wesley, 4 edition, April 2004.
- [Ste95] C Stephanidis. Towards user interfaces for all: Some critical issues. panel session "user interfaces for all-everybody, everywhere, and anytime". In *Symbiosis of Human and Artifact-Future Computing and Design for Human-Computer Interaction, Proceedings of the 6th International Conference on Human-Computer Interaction (HCI International '95)*, pages 137–142. Elsevier, Elsevier Science, 1995.
- [TCC⁺10] Giovanni Tummarello, Richard Cyganiak, Michele Catasta, Szymon Danielczyk, Renaud Delbru, and Stefan Decker. Sig.ma: Live views on the web of data. *J. Web Sem.*, 8(4):355–364, 2010.
- [TDO07] Giovanni Tummarello, Renaud Delbru, and Eyal Oren. Sindice.com: weaving the open linked data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference, ISWC'07/ASWC'07*, pages 552–565, Berlin, Heidelberg, 2007. Springer-Verlag.
- [Ted08] D. Tedesco. *Site Map Usability: 47 Design Guidelines Based on Usability Studies with People Using Site Maps*. Nielsen Norman Group, 2008.
- [Thi90] Harold Thimbleby. *User Interface Design*. Addison-Wesley, 1990.
- [Tho07] Jim Thomas. Visual analytics: a grand challenge in science: turning information overload into the opportunity of the decade. *J. Comput. Sci. Coll.*, 23(2):5–6, December 2007.
- [Tid05] Jenifer Tidwell. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly, 2005.
- [TJ92] David Turo and Brian Johnson. Improving the visualization of hierarchies with treemaps: design issues and experimentation. In *Proceedings of the 3rd conference on Visualization '92, VIS '92*, pages 124–131, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.
- [UHAS12] Jörg Unbehauen, Sebastian Hellmann, Sören Auer, and Claus Stadler. Knowledge extraction from structured sources. In Stefano Ceri and Marco Brambilla, editors, *SeCO Book*, volume 7538 of *Lecture Notes in Computer Science*, pages 34–52. Springer, 2012.
- [VFTjH05] Adapting Graph Visualization, Flavius Frasinca, Ru Telea, and Geert jan Houben. Adapting graph visualization techniques for the visualization of rdf data. In *of RDF data, Visualizing the Semantic Web, 2006*, pages 154–171, 2005.

BIBLIOGRAPHY

- [vHvW02] Frank van Ham and Jarke J. van Wijk. Beamtrees : Compact Visualization of Large Hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, volume 00, Los Alamitos, CA, USA, 2002. IEEE Computer Society.
- [W06] Holger Bast 0001 and Ingmar Weber. Type less, find more: fast autocompletion search with a succinct index. In Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, and Kalervo Järvelin, editors, *SIGIR*, pages 364–371. ACM, 2006.
- [W3C] World Wide Web Consortium W3C. W3c semantic web activity. <http://www.w3.org/2001/sw/>.
- [wai] Web accessibility initiative (wai). <http://www.w3.org/WAI/>.
- [WAs08] Max L. Wilson, Paul André, and mc schraefel. Backward highlighting: enhancing faceted search. In *Proceedings of the 21st annual ACM symposium on User interface software and technology, UIST '08*, pages 235–238, New York, NY, USA, 2008. ACM.
- [WksS10] Max L. Wilson, Bill Kules, m. c. schraefel, and Ben Shneiderman. From keyword search to exploration: Designing future search interfaces for the web. *Found. Trends Web Sci.*, 2(1):1–97, January 2010.
- [WoCloCS92] C. Wharton and University of Colorado. Institute of Cognitive Science. *Cognitive Walkthroughs: Instructions, Forms and Examples*. Institute of Cognitive Science, 1992.
- [WP06] Taowei David Wang and Bijan Parsia. Cropcircles: Topology sensitive visualization of owl class hierarchies. In *In Proc. of the 5th International Conference on Semantic Web*, pages 695–708, 2006.
- [YSLH03] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI conference on Human factors in computing systems, CHI '03*, pages 401–408, New York, NY, USA, 2003. ACM.