AN EXPERT-SUPPORTED APPROACH TO DATA EXPLORATION

A thesis submitted to the

University of Dublin, Trinity College

for the degree of

Doctor of Philosophy

Cormac Hampson
Knowledge and Data Engineering Group (KDEG)
School of Computer Science
Trinity College, Dublin,
Cormac.Hampson@tcd.ie

Supervised by Dr. Owen Conlan

Declaration

I, the undersigned, declare that this work has not been previously submitted as an exercise for a degree at this or any other University, and that, unless otherwise stated, it is entirely my own work.

Cormac Hampson
May 2011

Permission to Lend or Copy

Acknowledgements

Many people have provided support to me during the course of my Ph.D. studies. Firstly, I would like to thank my supervisor Dr. Owen Conlan for his guidance and many helpful contributions to this research over the years. I would also like to express my gratitude to my colleagues in KDEG who have been so generous with their time since I joined Trinity College.

A special mention must go to Dr. Rachael Rafter and Dr. Alex O'Connor for their time and feedback while I was compiling this thesis. Furthermore, I would like to thank Meltem Gürel, Thomas Hengster, Virgile Brouard and Aonghus McGovern, whose work with various incarnations of SARA and SABer helped to shape their implementation greatly. Many thanks are also due to Peter Williams for helping to initiate this research path many moons ago.

I would also like to acknowledge my sponsors who funded this research: The Embark Initiative of the Irish Research Council for Science, Engineering and Technology, funded by the National Development Plan; and Science Foundation Ireland via grant 08/IN.1/I2103 — Adaptive and adaptable media and services for dynamic personalisation and contextualisation (AMAS).

Finally, I would like to take this opportunity to express my gratitude to my family and friends for their encouragement throughout this entire process, and especially to Judy for her boundless patience and good humour.

Knowledge is knowing that a tomato is a fruit; wisdom is knowing not to put it in a fruit salad.

Abstract

Almost all information domains have witnessed a large increase in the amount of structured and semi-structured data available. However, there is still a lack of support for casual computer users who wish to create queries spanning multiple information sources. Until this occurs, the real benefits of having such a proliferation of metadata will not be realised by the general public. This thesis proposes a novel *expert-supported approach to data exploration* that will help casual users interact with large content repositories. Specifically, this approach helps users leverage expert knowledge to discover relevant information and to draw correlations across separate data sources. These heterogeneous sources can be in various data formats, and are accessed by users in a consolidated fashion.

Both a framework and a set of models based on this approach have been designed and are implemented in a technical infrastructure called SARA (Semantic Attribute Reconciliation Architecture). An associated authoring tool called SABer (Semantic Attribute Builder), which works in tandem with SARA has also been developed, and provides support for domain experts with no computing programming or data modelling experience to encode their expertise. Importantly, this means that SARA can be used in a broad range of domains, once rich metadata is available. How this expertise can then be tailored to an end-user's interpretation or context, in order to provide him with more meaningful semantics, is another key issue tackled in this research.

In summary, this thesis presents a novel and generic knowledge access platform that serves as an intermediary between curators and consumers of data. It describes the *expert-supported approach to data exploration*, its accompanying framework and models, as well as the implementation of SARA and SABer. Furthermore, the validation of these systems and their underlying approach is performed through five distinct evaluations. These evaluations incorporate a variety of techniques, including user trials, performance tests, questionnaires and interviews, and involve experiments with both SABer and SARA, and the third party applications that use them.

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
CNL	Controlled Natural Language
GUI	Graphical User Interface
HCIR	Human Computer Information Retrieval
IR	Information Retrieval
NLI	Natural Language Interface
OWL	Web Ontology Language
KDDM	Knowledge Discovery & Data Mining
RDF	Resource Description Framework
SABer	Semantic Attribute Builder
SARA	Semantic Attribute Reconciliation Architecture
SD	Standard Deviation
SME	Subject Matter Expertise
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SUS	System Usability Scale
URI	Uniform Resource Identifier
XML	Extensible Markup Language
XQuery	XML Query Language
XSL	Extensible Stylesheet Language

Abbreviations for Implementation Features

F1	Support communication with different data sources of various formats in a given domain.
F2	Store SME encoded by non-technical experts.
F3	Advertise this SME to third party applications, via an API that only requires parameters, so that end users can leverage the SME in their data explorations.
F4	Support extensibility and reusability of SME and data source registration, as well as domain independence.
F5	Enable end users to send complex, tailored queries based on semantic attributes, via client applications that send the associated parameters to an API.
F6	Allow individual queries encapsulated within the Semantic Attribute Models to be forwarded to the relevant data sources in the appropriate query language.

F7	Reconcile results from the separate data sources at an instance level.

F8	Return a consolidated set of results to the client application for rendering to the user.

Abbreviations for Evaluation Criteria

E1	SARA provides client applications with access to data sources without prescribing a specific user interaction paradigm for the GUI, or for their developers to know the underlying query language associated with each source.
E2	SABer enables the SME leveraged by the client applications to be encoded by non-technical experts.
E3	Casual users that appropriate (or tailor) this SME within a client application can send complex queries to data sources via SARA.
E4	SARA and SABer are domain independent.
C1	Technical features of SARA showcased in music case study.
P1	Performance evaluation of SARA to see under what circumstances it is a usable middleware.

1 Introduction

1.1 Motivation

In recent years, casual computer users have found themselves accessing diverse structured and semi-structured data on an increasingly regular basis. A casual[1] computer user may be defined as one with Internet browsing ability, but without programming skills or data modelling expertise (Huynh et al. 2008). With the proliferation of web services and mash-ups, and the advent of the Linking Open Data community project[2], this trend of casual users interacting more and more with distributed data sources is likely to continue. Even within single enterprises and organisations, it is not unusual for casual users to have access to many separate databases that store related information in different formats and schemas. However, while one may intuitively expect that any additional structure in the data would be exploited to provide sophisticated query capabilities, this has largely not proven to be the case (Bizer et al. 2009). Many applications that use structured data do provide access to their underlying data store via query languages; however these are suitable primarily for application developers with a knowledge of the language rather than casual end-users wishing to ask very specific questions through a usable human interface (Bizer et al. 2009). The importance of such functionality has already been identified in human-centred computing, with one of its main aims being to support users in making queries and seeing responses in their own terminology (Kurgan & Musilek 2006).

KDDM (Knowledge Discovery and Data Mining) techniques have increasingly found a niche in research and commercial environments (Mannila & Gunopulos 2009), enabling users to find interesting relationships and trends within large data collections (both structured and unstructured), despite these attributes not being explicitly encoded. Popular KDDM approaches include those based on statistical, case-based, neural network and probability techniques (Goebel & Gruenwald 1999). However, due to the complexity of these formal methods of knowledge seeking, users are often unwilling to devote the time and resources needed to learn them (Rouse 2003). Though KDDM techniques have proven to be very useful to specialists in a given discipline, their complexity means that they are

[1] In this thesis the term "casual" only refers to users' degree of computer expertise and not to their level of interest in interacting with and learning more about specific domains.
[2] http://esw.w3.org/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

not suitable for supporting users with limited computer skills in correlating useful information from separate data sources.

Data exploration techniques on the other hand, which are related to data mining (since they are concerned with exposing patterns and relationships in data), generally take a more human-centred approach to pattern discovery (Scianta Intelligence 2005). In contrast to KDDM which relies on automated algorithms (typically guided by human-specified parameters), data exploration explicitly incorporates manual techniques to support a user in locating important aspects of data sets. These manual techniques typically involve end users browsing and querying data sets through a GUI, in order to retrieve and examine data in an interactive and intuitive-based process of trial and error.

Imagine a casual computer user trying to locate *"all nostalgia music artists currently touring the USA"*. The information necessary to answer this rather subjective query is likely to be stored over multiple sources, e.g. one data source might contain information on upcoming concerts that are scheduled, and another might contain music chart data classified by artist and genre. In order to solve the user's query, the information from both sources must be consolidated, and therefore there is a clear need for techniques that support the casual user in this manner. One such way to support casual users in exploring data sets is to give them access to Subject Matter Expertise (SME). Such human domain expertise is widely used in Expert Systems, which are developed to help users find reliable information in narrow areas such as medicine or accounting (Vaughan-Nichols 2006). By leveraging and tailoring domain expertise, a user with limited knowledge of the music domain can be helped to search for *"all music artists that have concerts scheduled in the USA, despite their most recent top ten album in the USA being more than ten years ago"*. Importantly, by supporting the user in inputting specific values (top *ten* albums, more than *ten* years ago etc.), this approach is likely to provide the user with results that agree with his own interpretation of a *"nostalgia music artist touring the USA"*. Thus the user is facilitated in combining and interacting with multiple concepts (some of which may be quite fuzzy) that have been formalised by domain experts[3], in order to locate and correlate relevant information. Other examples of the types of query this approach supports include:

- finding all good value Asian restaurants that are near to my house.

[3] In terms of this thesis, an "expert" is defined as anyone with an understanding of a specific domain and the ability to express their knowledge. Approaches should enable SME from a specific expert, or group of experts, to be leveraged by casual users.

- locating all European holiday destinations serviced by low cost carriers, that have hot weather over Christmas, and an availability of rooms in luxury hotels.
- getting all patients that have a high risk of cardiac problems, which have not had a recent check up.
- listing all American universities that are highly ranked internationally, run a journalism M.Sc. course, and has cheap accommodation on campus.

What all these queries have in common, despite being in different domains, is that they combine data from different sources to satisfy a specific information need. Supporting casual users to explore multiple data collections using SME can also help them to expose useful relationships not immediately clear from examining the data sources individually. For example the increasing availability of government health, transport and education data can open up the opportunity to discover bicycle accident black spots and correlations between certain environmental factors and high levels of a disease (Hall 2010). Unfortunately the complexity of encoding SME into many expert systems (the comprehensive development of an ontology and its corresponding inference engine rules), is such that a non-technical[4] domain expert is likely to require the support of a knowledge engineer. This in turn increases the time and costs involved in such an exercise. Furthermore, the SME encoded often resides in a standalone system with little scope for expertise to be reused in different applications. This is especially the case when the user interface, domain expertise, and knowledge base are tightly coupled together. If all these components are unnecessarily bound together in one system, it is less likely that useful GUIs, rich information sources and carefully constructed SME will be reused elsewhere resulting in an undesirable duplication of effort.

The use of First Order Logic in SME also imposes complex rules that need to be understood by domain experts and the generation of inference rules may quickly become intractable. These are major challenges for non-technical domain experts to overcome, and these experts would require the support of a knowledge engineer to facilitate them. Interestingly, it has also been argued that some knowledge representation techniques seem to insist on solving harder problems than the user actually has (Rector 2006). Hence a

[4] A non-technical domain expert is one who has no formal background in computer programming or data modelling.

simpler more flexible approach to encoding SME may actually be just as useful as traditional methods of knowledge representation, such as ontologies and inference rules.

In terms of facilitating interesting correlations and trends to be elicited by users, consolidated access to multiple data sources (even within a single domain) is a crucial step forward towards more effective data management. Hence, many of the benefits offered by structured and semi-structured data will not be fully realised by users until such data integration becomes more widespread. Unfortunately, by its very nature, the structured data on the Internet is very heterogeneous and current data integration architectures cannot cope with this web-scale heterogeneity (Madhavan et al. 2007). This situation has seen an increase in *pay as you go* integration inspired by dataspaces (Halevy et al. 2006), where it is not necessary to have all data sources in a dataspace tightly integrated from the very beginning (e.g. free text search could be offered on a particular source before its schema is mapped). Furthermore, with the rapid growth of Web Services and Linked Data there has been an increasing interest in the lightweight integration of data sources (Bizer et al. 2009). In terms of Web Services, this has led to the proliferation of mash-ups and aggregated feeds. However, of even more potential is the use of dereferenceable URIs (Uniform Resource Identifiers) in Linked Data (Mei et al. 2008) which push integration down to the instance level. This means that each Linked Data object (person, place, product etc.) has an unambiguous reference, like a database id, and can be correlated with other references to it in multiple distributed data sources. This relatively recent approach facilitates more agile data integration, and is potentially much more powerful than traditional methods such as schema mapping.

As has been discussed, there is a clear need for techniques to support casual users in browsing, querying and drawing correlations within multiple sources from a domain. KDDM techniques appear too complex for casual users, with data exploration approaches seen as a more suitable alternative. SME has been successfully employed to support users in exploring specific domains. However, often the complexity of this SME requires the help of knowledge engineers to be encoded, which can greatly increase time and costs. Hence, the focus of this thesis is on supporting data exploration of multiple data sources by casual users, through SME encoded by non-technical domain experts. As the prevalence of and dependence on digital data escalates, and the increasing use of such repositories by casual users occurs, the need for user-friendly systems capable of supporting meaningful exploration will be greatly increased.

1.2 Research Question

This research asks how Subject Matter Expertise may be effectively encoded by non-technical experts and then leveraged by casual users to assist exploration and querying of multiple data sources from a domain.

This thesis describes an innovative approach, and its accompanying framework and models, which supports casual users in engaging with data in a more meaningful manner. This approach can support both systematic data exploration in a professional context, as well as more ad hoc explorations by users wishing to find out more about a specific domain. A major aspect of the framework described is its novel authoring tool, which enables non-technical experts to translate abstract concepts and qualitative information from their domain into concrete rules. These rules are then encapsulated within an SME model, and made available for use and manipulation by casual end users engaged in data exploration.

Non-technical experts and casual users both only have basic computing skills such as Internet browsing, and have no formal background in computer programming or data modelling. However, they differ through their degree of expertise in the domain. By "effectively encoded" it means that the representation of SME can be accurately transformed into a query language that accesses the raw data sources, and that the SME authoring tool is usable by non-technical experts. In the context of this thesis, a usable tool is one that follows the ISO definition of usability (Jokela et al. 2003), which uses effectiveness, efficiency and satisfaction as metrics.

1.3 Research Objectives

1) Analyse the state of the art in data exploration to determine the extent to which casual users are facilitated, and examine the state of the art in SME encoding for non-technical experts to identify the main features of current approaches.
2) Define an approach that allows end users to leverage SME (tailoring as appropriate) when exploring and consolidating information from separate data sources in a domain, and describe the accompanying models and framework necessary to implement it.
3) Perform evaluation studies to assess:
    a) the usability (effectiveness, efficiency and satisfaction) (Jokela et al. 2003), of the implemented SME authoring component of the framework, and the ability of non-technical users to encode SME.

b) whether the encoded SME can be usefully exploited by client applications to adequately support end-user exploration.

c) the features of the framework implementation and whether the consolidation of data from separate sources is supported.

1.4 Contribution

This thesis proposes an innovative solution that incorporates Subject Matter Expertise (SME) into the exploration of heterogeneous data sources. The primary motivation of this approach is to use SME, encoded by non-technical experts, to help casual users explore, query and correlate the data they encounter on a daily basis. The major contribution of this thesis is the *expert-supported approach to data exploration* and its accompanying models that support it. Importantly, this approach specifies a novel and generic knowledge access platform, which serves as a useful intermediary between curators and consumers of data. Specifically it contributes to the state of art through its approach to supporting non-technical experts in defining rules relating to notional or abstract concepts in their domain, which can then be leveraged and manipulated by end users to explore information in which they are interested in. The encoding of SME in this way also gives end users a common frame of reference in which to view their domain of interest.

The expert rules are encoded in a novel reusable SME model that contributes to the state of the art by natively supporting users who wish to tailor these expert rules to their own interpretation or context, which is often necessary because SME sometimes encompasses rather subjective notions such as “cheap” or “near”. Furthermore, by targeting domain experts with limited computer and data modelling skills, the approach proposed in this thesis is attractive to a wide audience of experts in a variety of domains, and its use of agile data integration techniques allows reusable dataspaces to be formed. These dataspaces help support the reconciliation of data from multiple sources in a domain. The approach presented in this thesis, if widely deployed online, has the potential to help systemise the appropriation of expert judgement by casual users, in order to assist meaningful exploration of a domain.

The minor contribution of this thesis is the framework to support the *expert-supported approach to data exploration*. The implementations of SARA (Semantic Attribute Reconciliation Architecture) and SABer (Semantic Attribute Builder) are instruments of this framework which were used to showcase its features. These systems have already been used to support two M.Sc. theses (Hengster 2009; Gürel 2009), two internship projects, and their details have appeared in several peer-reviewed publications (Hampson 2007; Hampson & Conlan 2009; Hampson & Conlan 2010; Hampson et al. 2011; Hampson & Conlan 2011). SARA and SABer have also recently been deployed as central technologies in the Science Foundation Ireland funded AMAS project[5] where they are being used by a multinational publishing house in order to support a consolidated visualisation of their vast e-learning resources (Hampson et al. 2011). Moreover, SARA has also been identified for deployment within the Science Foundation Ireland funded FAME project[6], where low level streaming data will be semantically enriched using SARA, in order to provide key information and alerts to a network management dashboard (Conlan et al. 2010). In summary, the successful deployment and evaluation of SARA and SABer validates the framework they represent, and reinforces the value of the *expert-supported approach to data exploration* and its models.

1.5 Technical Approach

An initial state of the art review was undertaken in the areas of *Data Exploration* and the *Encoding of SME*. After analysing the results of this broader review, a more focussed survey was conducted on *Complex Querying by Casual Users over Multiple Sources* and *Domain Independent Tools to Support SME Encoding by Non-Technical Experts*. *Complex Querying by Casual Users over Multiple Sources* specifically pertains to systems that provide casual users with more sophisticated mechanisms with which to make queries over structured and semi-structured data than would be possible with a more naïve key word search. Likewise *Domain Independent Tools to Support SME Encoding by Non-Technical Experts* examines generic systems that can be repurposed to different domains.

The literature review helped in providing an understanding of the major issues in these fields and in identifying the strengths and weaknesses of current approaches. Based on the state of the art analysis, a design for the *expert-supported approach to data exploration* was developed, which consists of seven distinct processes. Furthermore, detailed specifications of the various framework components, interfaces and models that are necessary to facilitate this approach were also generated. The models created include the

[5] http://kdeg.cs.tcd.ie/amas
[6] http://www.fame.ie/

Semantic Attribute Model and the Source Model, both of which are central to encoding SME in a flexible manner and enabling the reconciliation of data from multiple sources.

The base requirements for each of the major components of the framework were then listed in order to guide the technical implementation of the framework. These requirements and guidelines were closely adhered to in the development of SARA (Semantic Attribute Reconciliation Architecture) and SABer (Semantic Attribute Builder), which were distinct implementations of the two major components of the framework. A multi-source case study of a SARA installation in the music domain was also undertaken to highlight many of the features, stemming from the underlying approach, that SARA offers client applications.

Five distinct user experiments were devised to evaluate SARA and SABer, and the underlying *expert-supported approach to data exploration*. These experiments incorporated a variety of techniques, including user trials, performance tests, questionnaires and interviews, and involved third party applications as well as SARA and SABer. Early experiments iteratively improved the underlying design and approach, as well as the technical implementations themselves. Further experiments highlighted how SARA successfully facilitates several client applications in leveraging domain expertise, and how users benefit from these applications' use of SARA. Finally, the performance of non-technical users when using SABer was compared to that of users with computer coding experience, and a technical evaluation of SARA detailed the performance levels that prospective application developers can expect.

1.6 Thesis Overview

Chapter 2 provides an analysis and review of the state of the art, both in terms of *Complex Querying by Casual Users over Multiple Sources* and in terms of *Domain Independent Tools to Support SME Encoding by Non-Technical Experts*. Chapter 3 then describes a design that fulfils the goals and objectives outlined in Chapter 1, and highlights how key findings from the state of the art analysis helped to influence the *expert-supported approach to data exploration,* as well as its accompanying framework and models. The two main components of the framework (the Reconciliation Engine and the SME Authoring Tool) are detailed, as well as their associated processes, interfaces and data models.

How these framework components are technically realised through SARA (Semantic Attribute Reconciliation Architecture) and SABer (Semantic Attribute Builder) respectively is described in Chapter 4. This chapter has two main sections, the first of which highlights how SARA satisfies the requirements of the Reconciliation Engine component, and details its underlying technologies and interfaces. The second section discusses the implementation of SABer and how it supports non-technical domain experts to encode semantic attributes. This two-step process is described for all types of semantic attributes and data types that SABer supports. Finally this chapter concludes with a case study of a SARA installation in the music domain.

Chapter 5 details the various evaluations of SARA and SABer, and the underlying *expert-supported approach to data exploration*. These evaluation studies incorporate a variety of techniques, such as user trials, performance tests, questionnaires and interviews, and include experiments with SABer, SARA and the third party applications that use them. The goals, setup and results of each of the five distinct experiments are described, as well as an overall summary detailing the significance of the results. Conclusions are drawn in Chapter 6, and a discussion of the research contributions made is provided. Finally, this chapter also looks to the future and comments on some potentially interesting directions for this research.

2 State of the Art

This chapter describes a state of the art analysis of *Complex Querying by Casual Users over Multiple Sources* and *Domain Independent Tools to Support SME (Subject Matter Expertise) Encoding by Non-Technical Experts*. These state of the art reviews give special attention to features of other systems that are of particular relevance to this thesis. The main purpose of the reviews is to find out what systems are currently employed in these research areas, as well as to investigate what deficiencies systems in these fields suffer from. Key findings from the state of the art analysis are then listed, which are used to inform the system design discussed in Chapter 3.

2.1 Introduction

As described in Section 1.1, casual computer users are often unwilling to devote the time and resources to learn the formal methods of knowledge seeking used in KDDM (Rouse 2003). Furthermore, data exploration approaches were suggested as more suitable for users with a non-computer programming or data modelling background, and it was highlighted how SME has been successfully employed to support users in exploring domains of information (Vaughan-Nichols 2006). However, often the encoding of this SME requires the support of knowledge engineers, which can increase the associated time and costs of generating the SME. Hence, the motivation of this thesis is on supporting data exploration by casual users through SME encoded by non-technical domain experts. As a result, two core areas related to this thesis are presented; *Complex Querying by Casual Users over Multiple Sources* and *Domain Independent Tools to Support SME Encoding by Non-Technical Experts*. Each of these topics has important implications for the design choice described in Chapter 3.

2.2 Complex Querying by Casual Users over Multiple Sources

In this section a number of case studies of complex querying systems are examined. The systems chosen typically have a high impact value within the literature and follow an approach that is related in some way to that which is adopted by this research. The set of systems included for review also aims to reflect the diversity in the literature in terms of data format and user interaction paradigm (some systems that operate over single data sources are also included due to their interesting features). In particular this section concentrates on systems that are reusable across different domains, and that give casual users a more sophisticated approach to forming queries (or browsing) over structured and semi-structured data, than would be possible using simple keyword matching techniques (e.g. this thesis considers that users who just input keywords into search engines as performing simple querying). Complex querying approaches use mechanisms such as faceted interfaces, Natural Language Interfaces and widgets, in order to support users in finding specific answers and entities, in comparison to the ranked list of documents typical returned by search engines. In the course of analysing complex querying systems for Research Objective 1 (Section 1.3), the following criteria with which to evaluate these systems were derived:

1. What data formats does it work with?
2. Does it work with multiple data formats?
3. Does it work over multiple data sources?
4. Does it support third party applications to be developed on top of it?
5. Does the system use SME?
6. How does the user input queries via the GUI?

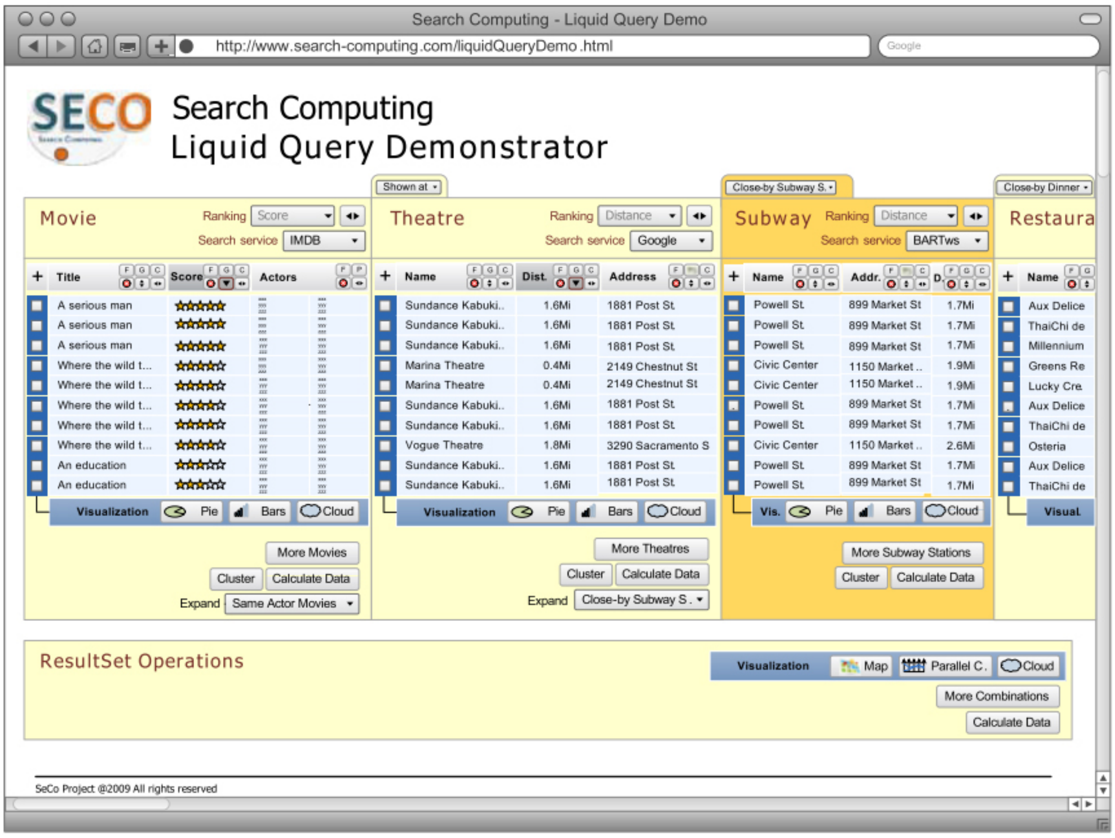
2.2.1 Search Computing Systems

Description:

Search Computing (Brambilla & Ceri 2009) defines a class of applications, which enable end users to perform exploratory search processes over multi-domain data sources on the Web. This research directly evolved from the New Generation Search framework (Braga, Ceri, et al. 2008; Braga, Calvanese, et al. 2008) that supported multi-domain queries over web services. It now encompasses an entire lifecycle from the initial registration of data sources to service marts, to the configuration of Liquid Query Templates, which are used to specify how end-users perform their exploratory searches in the GUI. More specifically, service marts are simple schemas which match “Web Objects” by hiding the underlying data source structures and presenting a simple interface (Campi et al. 2010). Likewise, Liquid Query Templates employ a set of input forms and controls to define user query submission, which are coupled with a set of parameters to specify result presentation (Bozzon, Brambilla, Ceri & Fraternali 2010).

Search Computing applications aim at filling the gap between general-purpose search engines, which are unable to find information spanning multiple topics, and domain-specific search systems, which cannot go beyond their domain limits. For instance a Search Computing application would support users in asking multi-domain queries such as

"Where can I attend a scientific conference close to a beautiful beach reachable with cheap flights?"(Brambilla & Ceri 2009). Search Computing proposes a software architecture that supports such multi-domain queries that span multiple sources. The distinguishing feature of Search Computing applications is the ability of combining, at query time, knowledge extracted from various domain-expert Web sources. This means that knowledge that is more accurate and complete than the knowledge available to general-purpose search systems is incorporated (Brambilla & Ceri 2009). Such expertise (about cultural events, medical specialisations, popular rock songs, and so on) is contributed through either social processes (e.g. rating, tagging, and commenting) or a careful knowledge construction process by experts.



Title	Score	Actors	
A serious man	★★★★★★	[illegible]	[illegible]
A serious man	★★★★★★	[illegible]	[illegible]
A serious man	★★★★★★	[illegible]	[illegible]
Where the wild t...	★★★★★☆	[illegible]	[illegible]
Where the wild t...	★★★★★☆	[illegible]	[illegible]
Where the wild t...	★★★★★☆	[illegible]	[illegible]
Where the wild t...	★★★★★☆	[illegible]	[illegible]
Where the wild t...	★★★★★☆	[illegible]	[illegible]
An education	★★★★☆☆	[illegible]	[illegible]
An education	★★★★☆☆	[illegible]	[illegible]

Name	Dist.	Address
Sundance Kabuki..	1.6Mi	1881 Post St
Sundance Kabuki..	1.6Mi	1881 Post St
Sundance Kabuki..	1.6Mi	1881 Post St
Marina Theatre	0.4Mi	2149 Chestnut St
Marina Theatre	0.4Mi	2149 Chestnut St
Sundance Kabuki..	1.6Mi	1881 Post St
Sundance Kabuki..	1.6Mi	1881 Post St
Vogue Theatre	1.8Mi	3290 Sacramento S
Sundance Kabuki..	1.6Mi	1881 Post St
Sundance Kabuki..	1.6Mi	1881 Post St

Name	Addr.	D.
Powell St	899 Market St	1.7Mi
Powell St	899 Market St	1.7Mi
Powell St	899 Market St	1.7Mi
Civic Center	1150 Market ..	1.9Mi
Civic Center	1150 Market ..	1.9Mi
Powell St	899 Market St	1.7Mi
Powell St	899 Market St	1.7Mi
Civic Center	1150 Market ..	2.6Mi
Powell St	899 Market St	1.7Mi
Powell St	899 Market St	1.7Mi

Name
Aux Delice
ThaiChi de
Millennium
Greens Re
Lucky Cre
Aux Delice
ThaiChi de
Osteria
Aux Delice
ThaiChi de

Figure 2-1. Mock Up of a Search Computing Application Result Page

Currently multi-domain queries can be answered only by patient and expert users, who interact with specialised engines one at a time. They then must feed the result of one search input to another one, reconciling answers manually. Although Search Computing is a promising field of research, it is important to note that this area is still "under construction" (Bozzon, Brambilla, Ceri, Corcoglioniti, et al. 2010), and as such there is a lack of evaluated applications that follow this paradigm.

Analysis:

Search Computing aims at supporting casual users who want to query and reconcile data from multiple data sources in different formats, which addresses a compelling need. One useful aspect of its GUI (specified in the Liquid Query Template) is that it offers the potential for complex queries to be easily generated by ordinary users. For instance simple widgets support the user in generating queries efficiently (e.g. sliding a bar representing the distance to a restaurant). Figure 2-1 shows a mock up result page (developed as part of the Search Computing Project) that details what a results page would look like in the user interface. The authors also discuss the possibility of incorporating Linked Data into the system which currently focuses only on web services. However, there are a number of potential issues with the current state of Search Computing, primarily stemming from its reliance on source providers to register (and configure) their services with service marts, and for experts to create Liquid Query Templates that specify the application GUI as well as how each widget interacts with the service marts. If both of these processes cannot be quickly and easily fulfilled, the likelihood of a wider adoption of this paradigm will decrease. Thus the simplification of these processes or the addition of tool support (especially in the generation of Liquid Query Templates) would be of great benefit. Furthermore, since each Liquid Query Template is designed for a specific application, direct reuse of these expert templates in other applications is not possible. Finally, though Search Computing shows significant potential in supporting complex multi-domain queries by casual users, it has been acknowledged by the developers that there is still a lack of user evaluation studies to validate the effectiveness and acceptance of the approach (Bozzon, Brambilla, Ceri & Fraternali 2010).

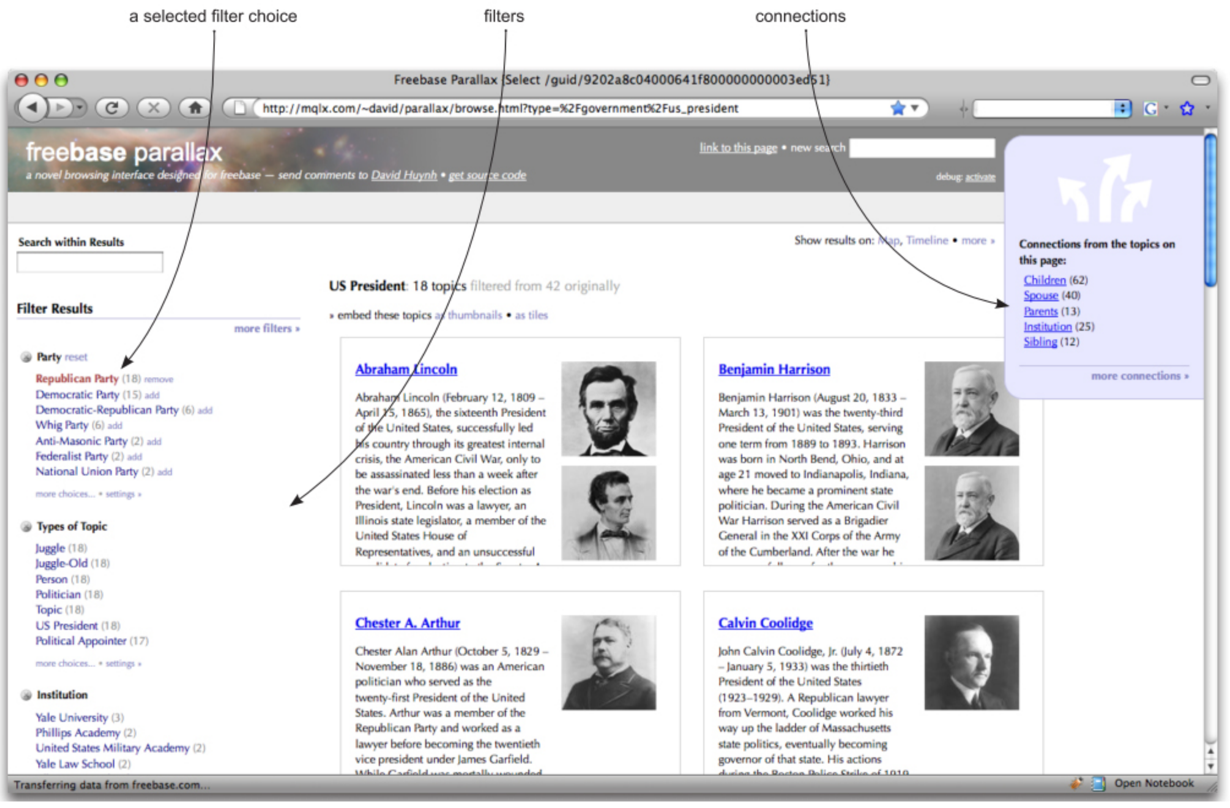
2.2.2 Parallax and Sparallax

Description:

Parallax (Huynh & Karger 2009) is an open source web application built on top of the Freebase[7] service, which allows people to use a set-based browsing paradigm to navigate through the structured data in Freebase (a diverse knowledge base composed mainly by its community members). The rationale for Parallax's set-based browsing paradigm is that dealing with one unit of web information at a time is insufficient for some information tasks. For instance, to get a basic understanding about the economy of Britain, one can simply read the corresponding Wikipedia article; however, if one wants to quantitatively

[7] http://www.freebase.com

compare the economies of various countries previously in the British colonial empire, then one must read many articles, extract out the quantities for comparison (e.g., Gross Domestic Product), tabulate them in a spreadsheet, and then construct a visualisation such as a bar chart to show the comparison. The developers postulate that in order to gain a big picture from bits and pieces extracted from several sources, current web browsers fall short due to their "one web page at a time" browsing paradigm, leaving the user to flip between several web pages and carry out tedious data tabulation manually (Huynh & Karger 2009).



Freebase Parallax {Select /guid/9202a8c04000641f800000000003ed51}

http://mqlx.com/~david/parallax/browse.html?type=%2Fgovernment%2Fus_president

freebase parallax

a novel browsing interface designed for freebase — send comments to David Huynh • get source code

link to this page • new search

debug: activate

Search within Results

Filter Results

more filters »

Party reset

Republican Party (18) remove
Democratic Party (15) add
Democratic-Republican Party (6) add
Whig Party (6) add
Anti-Masonic Party (2) add
Federalist Party (2) add
National Union Party (2) add

more choices... • settings »

Types of Topic

Juggle (18)
Juggle-Old (18)
Person (18)
Politician (18)
Topic (18)
US President (18)
Political Appointer (17)

more choices... • settings »

Institution

Yale University (3)
Phillips Academy (2)
United States Military Academy (2)
Yale Law School (2)

US President: 18 topics filtered from 42 originally

» embed these topics as thumbnails • as tiles

Show results on: Map, Timeline • more »

Abraham Lincoln

Abraham Lincoln (February 12, 1809 – April 15, 1865), the sixteenth President of the United States, successfully led his country through its greatest internal crisis, the American Civil War, only to be assassinated less than a week after the war's end. Before his election as President, Lincoln was a lawyer, an Illinois state legislator, a member of the United States House of Representatives, and an unsuccessful

Benjamin Harrison

Benjamin Harrison (August 20, 1833 – March 13, 1901) was the twenty-third President of the United States, serving one term from 1889 to 1893. Harrison was born in North Bend, Ohio, and at age 21 moved to Indianapolis, Indiana, where he became a prominent state politician. During the American Civil War Harrison served as a Brigadier General in the XXI Corps of the Army of the Cumberland. After the war he

Chester A. Arthur

Chester Alan Arthur (October 5, 1829 – November 18, 1886) was an American politician who served as the twenty-first President of the United States. Arthur was a member of the Republican Party and worked as a lawyer before becoming the twentieth vice president under James Garfield.

Calvin Coolidge

John Calvin Coolidge, Jr. (July 4, 1872 – January 5, 1933) was the thirtieth President of the United States (1923–1929). A Republican lawyer from Vermont, Coolidge worked his way up the ladder of Massachusetts state politics, eventually becoming governor of that state. His actions

Connections from the topics on this page:

Children (62)
Spouse (40)
Parents (13)
Institution (25)
Sibling (12)

more connections »

Transferring data from freebase.com...

Open Notebook

Figure 2-2. Parallax Interface Listing US Presidents

A user starts in Parallax by performing a keyword search, with Parallax suggesting items whose names match. When the user selects one of these suggestions, several related facets are auto-generated and displayed so that the user can narrow the results rendered further. Alongside the facets displayed in the interface, a list of connections is also presented. A connection is a list of links from the current result set to other database entries, defined by a particular relationship. For example, if there were entries on US Presidents displayed, a user could select the suggested *offspring* collection which would then switch Parallax from showing all the presidents in its database, to showing all the *offspring* of all the presidents in the database. This feature of set-based browsing lets the user move from one set of things to another related set of things. Hence by combining facets with collections it is possible to use Parallax to locate items such as *any offspring of all the Republican US Presidents who were Presbyterian.*

Analysis:

One of the major limitations of Parallax is that it is limited to connecting to the Freebase knowledge base (though it is a repository of considerable size). To broaden the applicability of this set based paradigm, Sparallax[8] was developed using Parallax in order to provide a faceted browsing interface for SPARQL endpoints. Currently it works with *DBpedia*[9] and *LinkedMDB*[10] SPARQL endpoints, though it is possible to specify the locations of other Linked Data repositories. Facets and collections are automatically generated as the user browses a data repository, however users of Sparallax are limited to searching one data source at a time, and it is not possible to reconcile information from different sources. Moreover, both Parallax and Sparallax only work with data sources of a single type (Freebase and RDF respectively). This issue may be mitigated by converting data sources to the formats supported or providing wrappers to them. However, this is an additional pre-processing step that would be unnecessary if multiple data sources were supported natively, and the conversion process itself may introduce new errors to the data sources. Finally there is no end user evaluation of either system published, so it is unclear yet if casual users fully accept the paradigm and can see its benefits. Typical of systems that work with distributed data sources, Sparallax can suffer from very high latencies which impinge on the browsing experience. However advances on scalable and efficient access to Linked Data will benefit the system in future, as this will reduce the number of times a user's interaction with the system is interrupted by lengthy delays.

[8] http://sparallax.deri.ie/
[9] http://dbpedia.org/sparql
[10] http://data.linkedmdb.org/sparql

2.2.3 Explorator

Description:

Explorator (De Araújo & Schwabe 2009) is a domain independent tool for exploring the Semantic Web, enabling users with minimal knowledge of RDF models to explore an RDF database without a priori knowledge of the domain. It aims at supporting a semantic web user in performing tasks such as finding *all papers mentioning another paper; or all paper authors' phone numbers*. In these tasks the user may encounter different data architectures, and information that is stored in multiple RDF files or expressed in distinct vocabularies. Hence Explorator aims at consolidating the data that is accessed in an integrated way and having the capacity to merge information described in different vocabularies. The developers of Explorator assert that current tools which allow the user to manipulate raw RDF data do not provide a user friendly way to ask questions, and that the user only has a limited way to rearrange, group or filter the data, and process it further.

In Explorator, every SPARQL endpoint is a repository that can be enabled or disabled, and can be manipulated individually or integrated into a single global source of RDF data. Users are also provided with a Query-by-example interface (Zloof 1975), as well as faceted navigation features. Similar to Parallax, Explorator uses a set-based paradigm, where the data resulting from an initial key word search (to match resources in RDF) can be further manipulated to either remove uninteresting elements or to add additional elements of interest. This manipulation includes the use of intersect, union and difference functions over various result sets in order to generate new sets of interest. This allows queries such as *finding all the lakes exclusively contained in Russia* to be resolved by getting the difference between set A (all the lakes in Russia) and set B (all the lakes in countries that have a border with Russia).

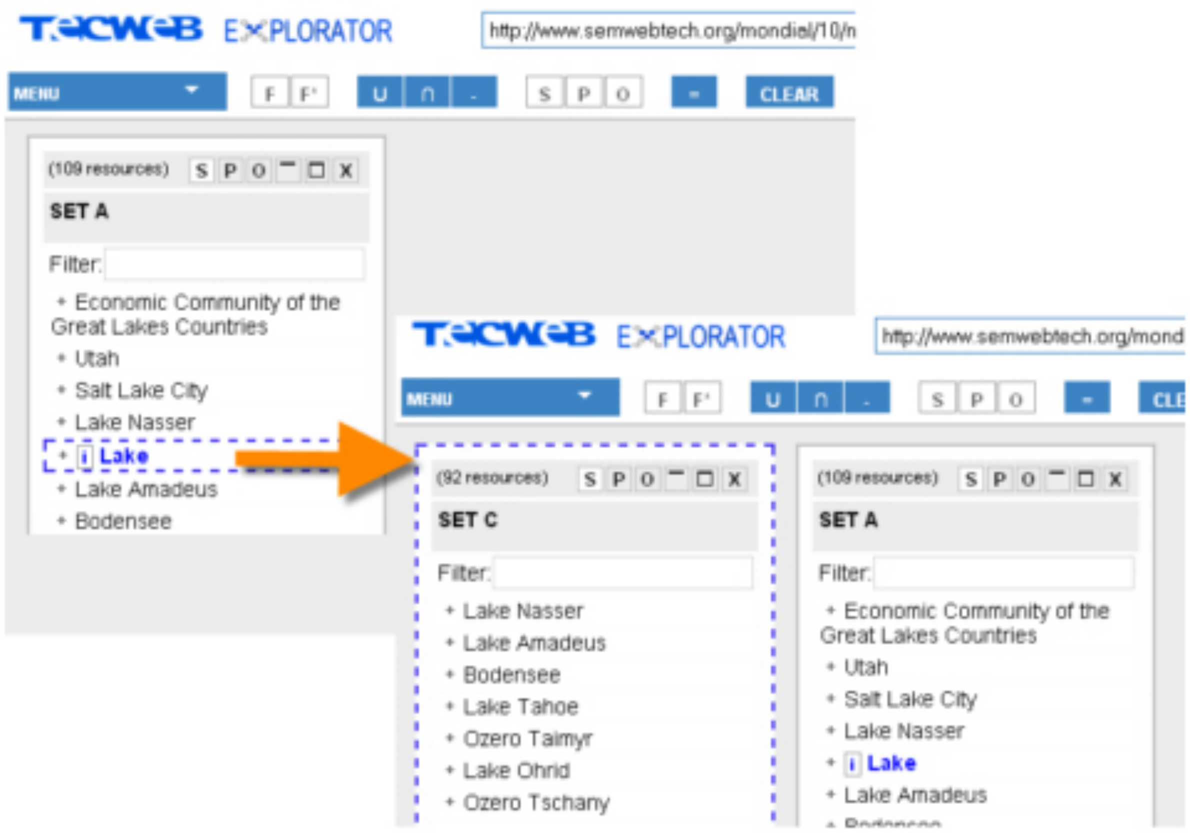


Figure 2-3. Explorator Interface Showing Different Sets

Analysis:

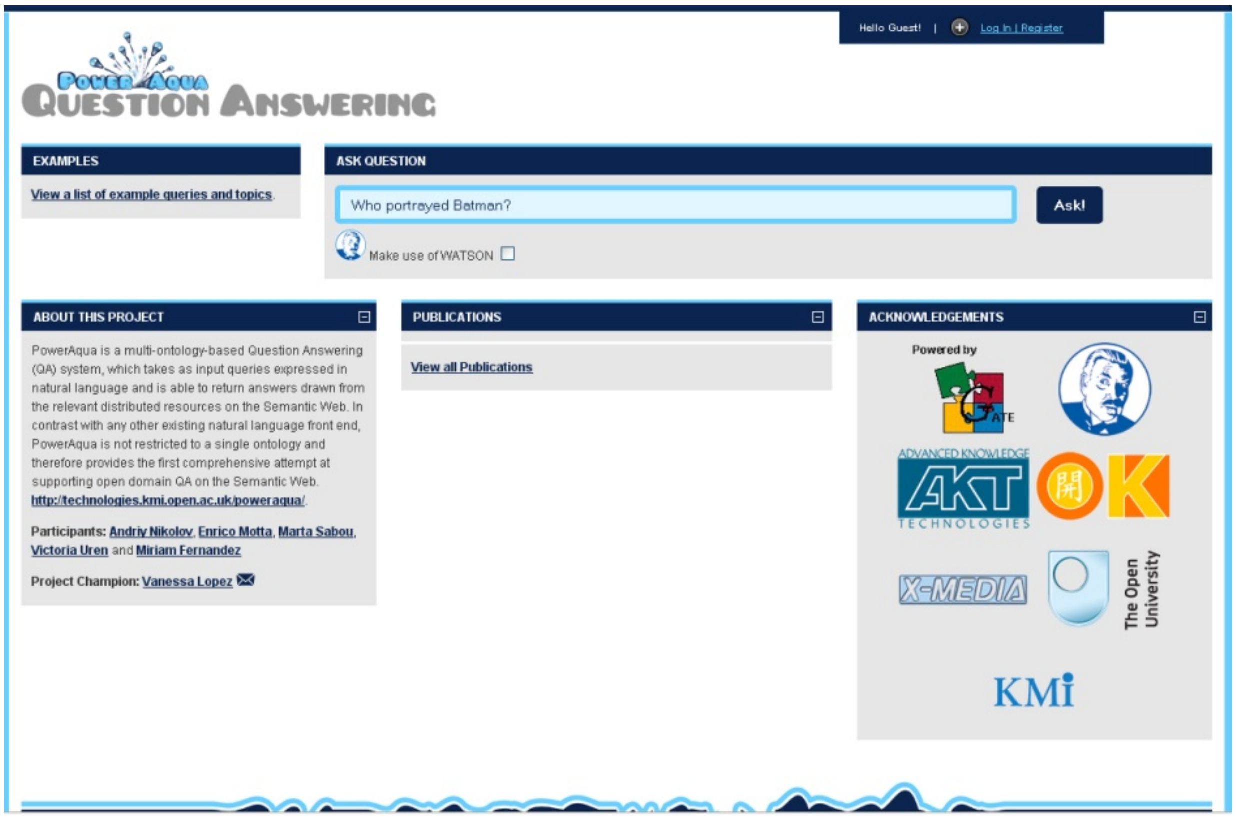
In terms of evaluating Explorator, initial small scale experiments have been conducted with users who knew some basic concepts of the semantic web and RDF, such as the use of subject/predicate/object triples (De Araújo et al. 2009). Despite the fact that these participants were more experienced than casual web users, they still struggled to perform tasks using Explorator. Specifically, they had problems exploring the domain because of the difficulty associated with finding appropriate properties to form queries. Furthermore, they had trouble interacting with the user interface in general, and found some visual elements unintuitive. The developers concluded that users with a basic knowledge of RDF were able to elaborate nontrivial queries with Explorator, however the user interface was a hindrance. From these results a redesign of the interface took place, however further evaluation studies will be necessary to validate this new design. In particular they are planning additional larger-scale experiments to compare alternative user interfaces and various interaction paradigms in order to better support both novice and expert users in exploring the Semantic Web. This is important as Explorator is currently better suited to advanced users who have solid knowledge of RDF.

There are clear advantages to system like Explorator, such as its domain independence and its potential to provide users with a powerful functionality such as set manipulation. It also uses dereferenceable URIs (Uniform Resource Identifiers) to identify resources (which helps to disambiguate terms), and the associated time cost of adding new RDF sources to the federated graph is low. One issue of note elicited through their experiments, is that user experience is hindered by the performance of the system when accessing remote sources. Similar to Sparallax, improvements in the speed of accessing remote repositories will benefit such systems enormously as it will reduce the number of times a user's interaction with the system is interrupted by lengthy delays.

2.2.4 PowerAqua

Description:

PowerAqua (Lopez et al. 2009) is an open multi-ontology Question Answering (QA) system for the Semantic Web developed from the earlier system AquaLog (Lopez et al. 2005). Users of PowerAqua input natural language queries and get answers from heterogeneous data repositories. In particular, PowerAqua is able to integrate, on the fly, statements drawn from different sources in order to generate integrated answers to questions. This is in contrast to many similar systems which limit their scope to a set of a priori selected medium size ontologies. Knowledge can be aggregated to complete information partially presented in single sources, fusing similar answers and filtering irrelevant ones. Furthermore, the most accurate answer(s), in terms of their relevance to the query and the varying levels of quality, popularity and trust, are elicited from different sources. PowerAqua has recently evolved to exploit the metadata offered by Linked Data, in order to answer linguistically simple questions such as "Find me university cities in Japan", which can require knowledge fusion across different sources. Initial experimental results regarding PowerAqua's performance over a subset of Linked Data (including DBpedia) are promising (Lopez et al. 2010), however the fact that relatively simple linguistic queries (e.g. *Give me airports in Canada, List me Asian countries*) took an average of 48.3 seconds to be processed, raises some efficiency concerns.



PowerAqua is a multi-ontology-based Question Answering (QA) system, which takes as input queries expressed in natural language and is able to return answers drawn from the relevant distributed resources on the Semantic Web. In contrast with any other existing natural language front end, PowerAqua is not restricted to a single ontology and therefore provides the first comprehensive attempt at supporting open domain QA on the Semantic Web. http://technologies.kmi.open.ac.uk/poweraqua/.

Participants: Andriy Nikolov, Enrico Motta, Marta Sabou, Victoria Uren and Miriam Fernandez

Project Champion: Vanessa Lopez

Figure 2-4. PowerAqua User Interface

Analysis:

Though PowerAqua only supports the RDF format, it does provide a consolidated access point to multiple data sources and reconciles results from heterogeneous sources. Furthermore, the setup costs are quite low as PowerAqua only requires access to indexed semantic documents via an API, and its users can start posing questions straight away. However, PowerAqua is currently a standalone website, and there is no facility for third party application developers to leverage the functionality of PowerAqua in their own systems.

PowerAqua's recent focus on Linked Data is another indicator on the growing importance of this format in this research area, and it is clear that any worthwhile query system should support access to repositories of this type. The developers of PowerAqua admit that in order for ordinary users to access Linked Data they have to be guided when building queries. Indeed they state that creating innovative ways to interact with Linked Data is crucial and even envisioned as a potential "killer app" (Lopez et al. 2010). Though the kind of Natural Language Interfaces (NLI) that systems like PowerAqua offer do give users a powerful freedom to express queries, this freedom often leads to the generation of questions that these systems are unable to answer correctly. Indeed Thompson (Thompson et al. 2005) suggests that we still cannot successfully use natural language to query and command computers, and to understand the difficulty, we should try using such a system as it won't understand many of your questions and commands. To illustrate with a simple example using PowerAqua, the query "What is the birthname of Bono?" yields the correct answer (Paul Hewson), the query "What is the birth name of Bono?" returns his wife's name (Ali Hewson), and "What is Bono's birth name?" finds no answers at all[11]. Moreover, it must be noted that these queries are quite simple and unambiguous, which plays to the strengths of NLI systems. More ambiguous or complex queries are likely to fare considerably worse. This is because words often have multiple meanings (lexical ambiguity) and a complex expression can have multiple underlying structures (structural ambiguity); queries with such ambiguous input often confuse NLI systems and lead to irrelevant search results (Hildebrand et al. 2007).

Such problems have led practitioners to propose that queries which cannot be answered by the search engine, due to inexistent ontological support, should not even be allowed, as the user becomes unsure if he has entered the wrong keywords or if the search engine simply does not have information to answer his search (Paiva & Ramos-Cabrer 2010). Until PowerAqua can provide a higher guarantee as to the accuracy of the answers it provides, such ambitious systems will struggle to gain wider acceptance. These general problems associated with natural language interfaces are discussed further in the following section on the ORAKEL system.

2.2.5 ORAKEL

Description:

Natural language is generally deemed to be the most intuitive form of querying from an end users point of view, however it has also been shown to be the most difficult to realise effectively. The main reasons for this difficulty are that:

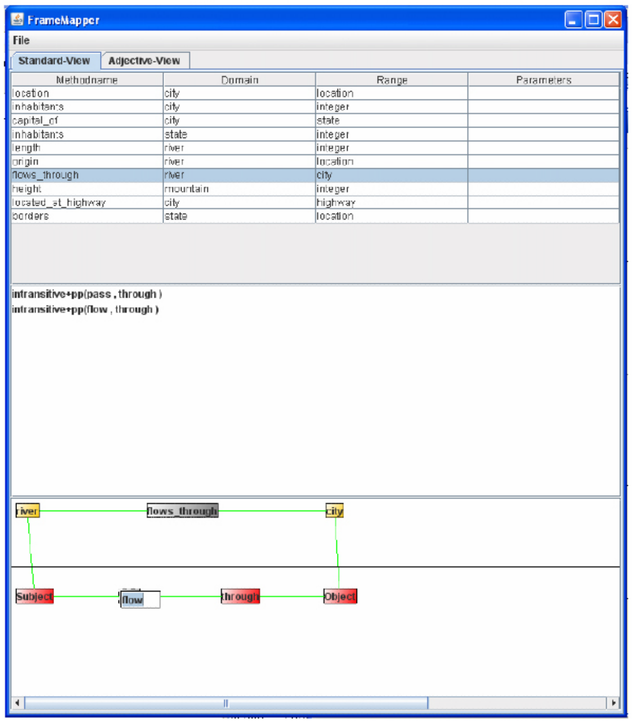
1. natural language understanding is a very difficult task due to ambiguities arising at all levels of analysis: morphological, lexical, syntactic, semantic, and pragmatic
2. a reasonably large grammar is required for the system to have an acceptable coverage
3. the natural language interface needs to be accurate

[11] These three indicative queries were inputted by the author on the live demo of PowerAqua located at http://poweraqua.open.ac.uk:8080/poweraqualinked/jsp/index.jsp

4. the system should be adaptable to various domains without a significant effort (Cimiano et al. 2008).

ORAKEL (Cimiano et al. 2008) aims to address these issues by supporting the customisation of Natural Language Interfaces (NLI) for a given domain, in order to provide more accurate results to user queries.

In ORAKEL there are two kinds of users, end users who send domain specific queries, and the domain experts or knowledge engineers who create domain-specific lexicons for adapting the system to a specific domain. A graphical tool called FrameMapper is provided which helps the domain experts with this process. Any inputted question is parsed by ORAKEL and a query in logical form is generated with respect to domain-specific predicates. This logical form is essentially a First Order Logic (FOL) representation enriched with query, count and arithmetic operators. The query in logical form is then translated into the target knowledge representation language of the knowledge base, in particular into its corresponding query language. The overall approach is thus independent of the specific target knowledge language, and can accommodate any reasonably expressive knowledge representation language with a corresponding query language. ORAKEL has thus far been tested with the F-Logic and OWL knowledge representation languages.



Methodname	Domain	Range	Parameters
location	city	location	
inhabitants	city	integer	
capital_of	city	state	
inhabitants	state	integer	
length	river	integer	
origin	river	location	
flows_through	river	city	
height	mountain	integer	
located_at_highway	city	highway	
borders	state	location	

intransitive+pp(pass , through)
intransitive+pp(flow , through)

Figure 2-5. FrameMapper Interface

Analysis:

Two case studies have been carried out with ORAKEL, the first using a small knowledge base containing geographic facts about Germany, and the second encompassing metadata about research publications from British Telecom's (BT's) digital library. The studies show that ORAKEL can be adapted by domain experts to different domains in a reasonable amount of time, typically a few hours. Although this task did not specifically require any computational linguistic knowledge, the users who performed it all had a computer science background, so whether the findings can be extended to casual computer users remains to be seen. PowerAqua does not require this manual step at all (relying on similarity metrics to make matches), however the developers of ORAKEL feel their approach yields more accurate results at a relatively low cost of domain customisation.

ORAKEL can work with knowledge representation forms of different kinds; however it can only work with a single knowledge base at a time. This limitation restricts its suitability for working with heterogeneous data sources from the Internet. Though ORAKEL has shown some promising results in terms of speed and accuracy (relative to similar NLI systems), the largest knowledge base evaluated only contained metadata for less than 120,000 items, so it has not been conclusively shown that the system scales to individual knowledge bases with billions of entities, which are becoming increasingly common. The coverage of the knowledge base that ORAKEL provides is directly related to the customisation step by the domain expert. Hence this stage is typically an iterative process with new mappings added over time in order to increase the range of questions that ORAKEL can accurately answer. With larger more heterogeneous knowledge bases this task may become very arduous.

2.2.6 Semantic Web Portal

Description:

The Semantic Web Portal (SWP) (Y. Ding et al. 2010) is a light-weight platform that unifies off-the-shelf Semantic Web tools to help domain users to organise, browse and visualise relevant semantic data in a meaningful manner. SWP is domain independent, and supports casual users in their exploration of multiple data sources through a single portal. These data sources are typically in different formats, although they must be transformed into RDF before inclusion to SWP. This process is a potentially time consuming task, and it may be challenging to ensure the conversions are done correctly. The main architecture of SWP itself is based upon the Longwell[12] faceted browser and the Exhibit[13] widget (creates web pages with advanced text search, filtering and visualisation functionalities) from MIT's SIMILE project[14].

Deploying SWP is domain specific, so a user needs to create one or more portal ontologies using the Ontology Management Component. This is an online tool based on Vitro[15] that enables easy creation, editing, browsing, mapping and annotation of ontologies. Creating an appropriate ontology is a critical part of SWP and should facilitate user queries, as well as the meaningfully display and visualisation of RDF data. The SWP developers claim that it allows non-Semantic Web users to create a new Semantic Web portal in a reasonable period of time without professional training, however there is no documented evaluation to support this claim. Indeed the generic requirements for creating portal ontologies appear to be beyond the scope of a casual web user with no professional training:

1. the ontology should reflect the database schema of its original datasets
2. the identified main concepts or relationships from commonly used user queries should be included in ontologies
3. to enable interoperability, the portal ontologies should try to reuse existing popular ontologies, such as using FOAF to represent people
4. Linked Open Data rules should be obeyed: using HTTP URIs for naming items, making URIs dereferenceable and trying to use URIs from other Linked Open Data as much as possible.

Once portal ontologies have been generated, users can then interact with the RDF data which can be either displayed as tiles or else visualised in a timeline, Google map or table formats. Users can also browse via facets or input keywords into its semantic search component.

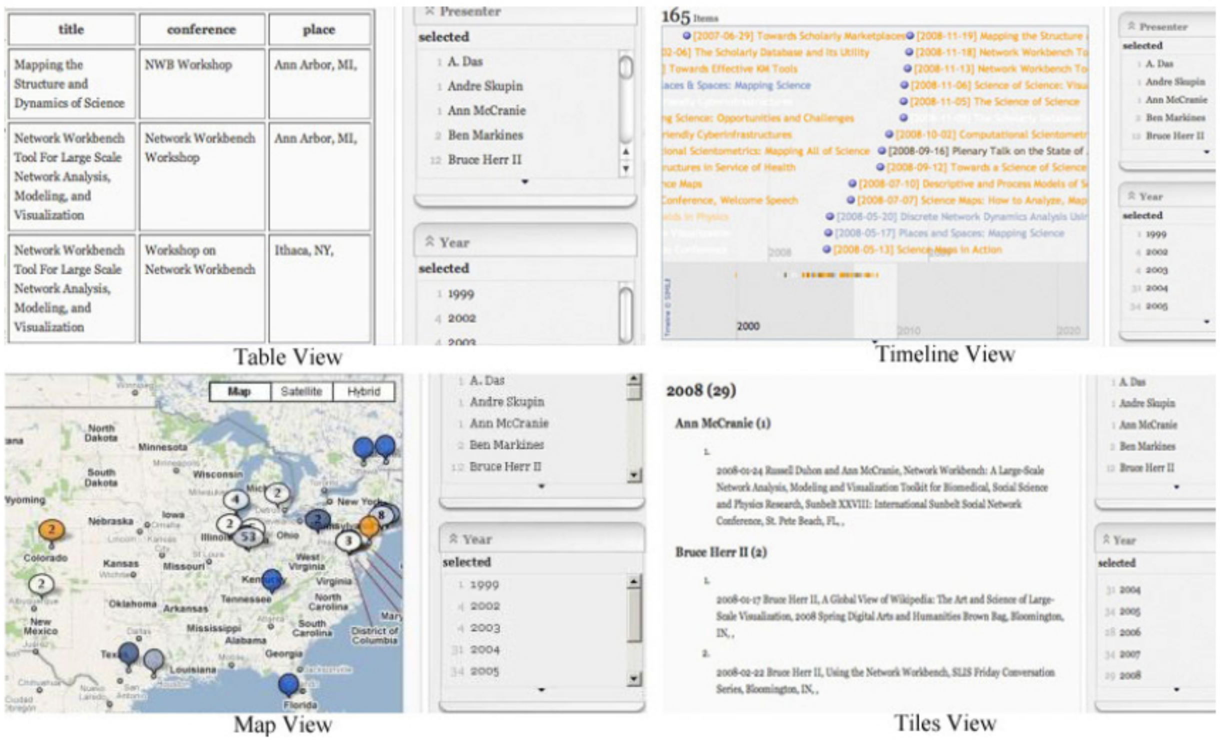
Analysis:

Currently SWP has been deployed to the following areas:

---

[12] http://simile.mit.edu/wiki/Longwell_User_Guide
[13] http:// simile-widgets.org/exhibit
[14] http://simile.mit.edu/
[15] http://vitro.mannlib.cornell.edu/

1. A research group (of 30 people) to semantically manage topics of people, paper, grant, project, presentation and research
2. A specialty Linked Open Data chem2bio2rdf dataset to display the relationship and association among gene, drug, medicine and pathway data.
3. An eGov dataset to facilitate faceted browsing of governmental data
4. A health centre to enable federated patient, disease, medication and family ties to be grouped, associated and networked.



title	conference	place
Mapping the Structure and Dynamics of Science	NWB Workshop	Ann Arbor, MI,
Network Workbench Tool For Large Scale Network Analysis, Modeling, and Visualization	Network Workbench Workshop	Ann Arbor, MI,
Network Workbench Tool For Large Scale Network Analysis, Modeling, and Visualization	Workshop on Network Workbench	Ithaca, NY,

Figure 2-6. Screenshots of SWP Visualisation component

However despite the potential shown in supporting third party applications, SWP does have some limitations. These include the necessity to convert all data sources to RDF, and the complicated way end users must add, delete and update their instance data. Furthermore, the generation of portal ontologies may be very time consuming, and there is no evidence that these portal ontologies can be generated without the help of knowledge engineers. Finally the developers make reference to SWP's applicability to "middle-sized domains" which suggests (without clarification as to what constitutes a "middle-sized domain") that the system may have issues with scalability.

2.2.7 MashArt

Description:

Mashups are web applications that are developed by integrating data, application logic, and user interfaces sourced from the Web. Mashup development is still an ad-hoc and time-consuming process, requiring advanced programming skills. The mashArt project (Daniel et al. 2010) aims at enabling even non-professional programmers to perform complex UI, application, and data integration tasks online in order to build sophisticated mashups. Although existing mashup approaches have produced promising results, techniques that cater for universal integration of web components at all the three layers of the application stack are still missing (Daniel et al. 2010). The developers feel that such techniques are necessary to transition Web 2.0 programming to an environment where users can leverage simple abstractions to create composite web applications out of professionally developed components.

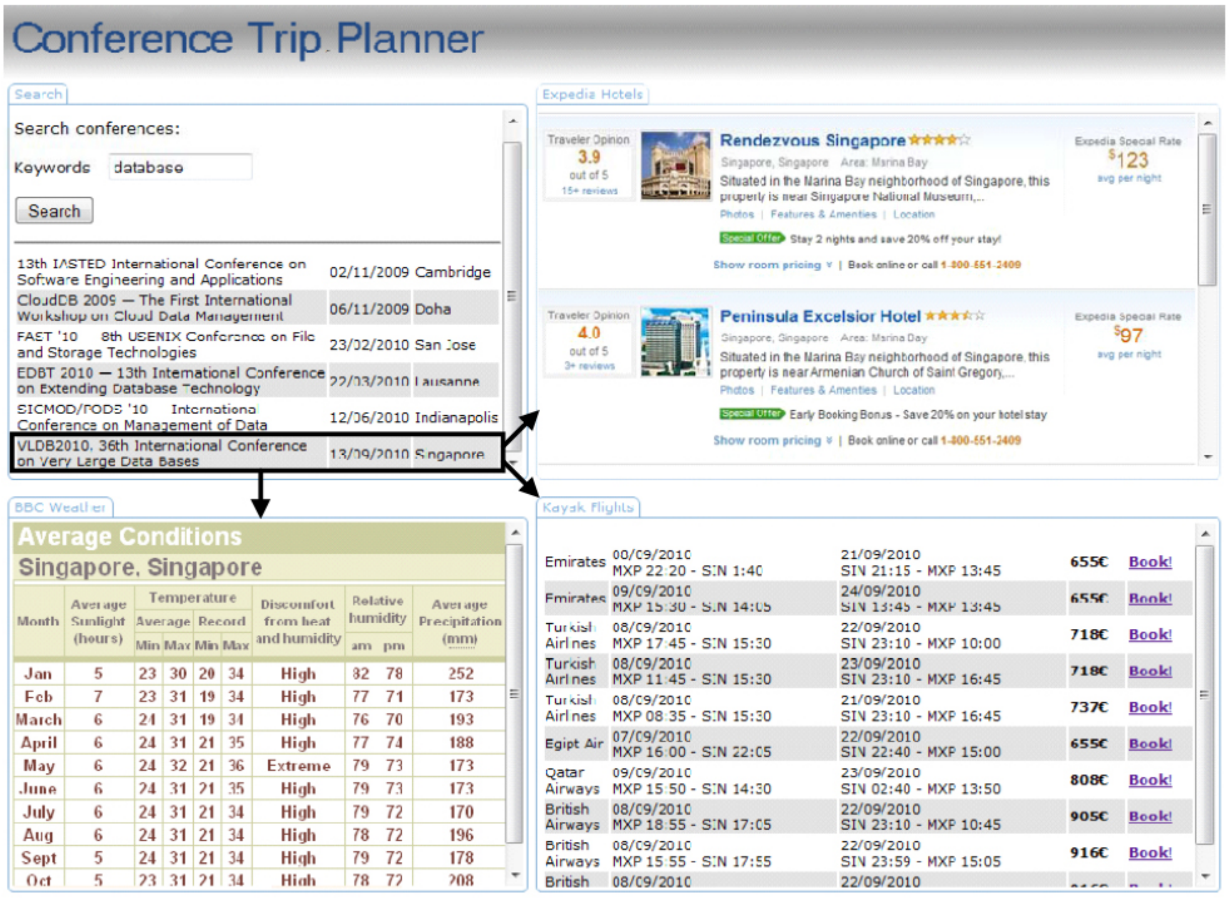
Technically mashArt consists of a graphical editor, a hosted execution environment, an online registry for components and compositions. Applications developed in mashArt consist of integrated data services (RSS, Atom feeds, JSON, XML etc.), web services and user interface elements. Hence, by integrating search results or services, mashups can become a natural candidate for the kind of search computing applications described earlier in Section 2.2.1, and indeed is a related project. For instance mashArt has been used to generate an application that allows web users to "*find all database conferences in the next six months in locations where the average temperature is 28°C degrees and for which a cheap travel solution including a luxury accommodation exists*" (Daniel et al. 2010). The mashArt graphical editor was used to compose such a Conference Trip Planner application. It consists of four integrated UI components; Conferences Search, Expedia Hotels, RSS Reader and BBC Weather, as well as two stateless service components; ConferencePipe and Kayak. The application has four listeners:

1. If a user enters a conference search string and starts the search, the ConferencePipe service is invoked by processing a Yahoo! pipe that queries two other services: conference-service.com and allconferences.com. The pipe joins the results coming from the two services and applies the filter condition provided by the user; the result is passed back to Conferences Search UI component.
2. If a user selects a conference from the list of retrieved conferences, three listeners reacting to the same event are activated.

a. The first listener propagates the selected conference location and dates to the Expedia Hotel service that retrieves a list of available hotels from the Expedia repository.

b. The second listener searches for matching flights and visualises them in the RSS Reader. The flights are retrieved by invoking a kayak.com flight search service and delivering its results as an RSS feed.

c. The third listener aligns the data shown in the BBC Weather component by forwarding the name of the city the conference is located in through the SearchWeather operation. This causes the component to visualise the average weather conditions for the selected city. (Daniel et al. 2010)


Conference Trip Planner

Search

Search conferences:

Keywords database

Search

13th IASTED International Conference on Software Engineering and Applications	02/11/2009	Cambridge
CloudDB 2009 — The First International Workshop on Cloud Data Management	06/11/2009	Doha
FAST '10 8th USENIX Conference on File and Storage Technologies	23/02/2010	San Jose
EDBT 2010 — 13th International Conference on Extending Database Technology	22/03/2010	Lausanne
SIGMOD/PODS '10 International Conference on Management of Data	12/06/2010	Indianapolis
VLDB2010. 36th International Conference on Very Large Data Bases	13/09/2010	Singapore

Expedia Hotels

Traveler Opinion 3.9 out of 5 15+ reviews

Rendezvous Singapore ★★★★☆

Singapore, Singapore Area: Marina Bay

Situated in the Marina Bay neighborhood of Singapore, this property is near Singapore National Museum,...

Photos | Features & Amenties | Location

Special Offer Stay 2 nights and save 20% off your stay!

Show room pricing | Book online or call 1-800-551-2409

Expedia Special Rate $123 avg per night

Traveler Opinion 4.0 out of 5 3+ reviews

Peninsula Excelsior Hotel ★★★½☆

Singapore, Singapore Area: Marina Bay

Situated in the Marina Bay neighborhood of Singapore, this property is near Armenian Church of Saint Gregory,...

Photos | Features & Amenties | Location

Special Offer Early Booking Bonus - Save 20% on your hotel stay

Show room pricing | Book online or call 1-800-551-2409

Expedia Special Rate $97 avg per night

BBC Weather

Average Conditions

Singapore, Singapore

Month	Average Sunlight (hours)	Temperature				Discomfort from heat and humidity	Relative humidity		Average Precipitation (mm)
		Average		Record					
		Min	Max	Min	Max		am	pm	
Jan	5	23	30	20	34	High	82	78	252
Feb	7	23	31	19	34	High	77	71	173
March	6	24	31	19	34	High	76	70	193
April	6	24	31	21	35	High	77	74	188
May	6	24	32	21	36	Extreme	79	73	173
June	6	24	31	21	35	High	79	73	173
July	6	24	31	21	34	High	79	72	170
Aug	6	24	31	21	34	High	78	72	196
Sept	5	24	31	21	34	High	79	72	178
Oct	5	23	31	21	34	High	78	72	208

Kayak Flights

Emirates	00/09/2010 MXP 22:20 - SIN 1:40	21/09/2010 SIN 21:15 - MXP 13:45	655€	Book!
Emirates	09/09/2010 MXP 15:30 - SIN 14:05	24/09/2010 SIN 13:45 - MXP 13:45	655€	Book!
Turkish Airlines	08/09/2010 MXP 17:45 - SIN 15:30	22/09/2010 SIN 23:10 - MXP 10:00	718€	Book!
Turkish Airlines	08/09/2010 MXP 11:45 - SIN 15:30	23/09/2010 SIN 23:10 - MXP 16:45	718€	Book!
Turkish Airlines	08/09/2010 MXP 08:35 - SIN 15:30	21/09/2010 SIN 23:10 - MXP 16:45	737€	Book!
Egipt Air	07/09/2010 MXP 16:00 - SIN 22:05	22/09/2010 SIN 22:40 - MXP 15:00	655€	Book!
Qatar Airways	09/09/2010 MXP 15:50 - SIN 14:30	23/09/2010 SIN 02:40 - MXP 13:50	808€	Book!
British Airways	08/09/2010 MXP 18:55 - SIN 17:05	22/09/2010 SIN 23:10 - MXP 10:45	905€	Book!
British Airways	08/09/2010 MXP 15:55 - SIN 17:55	22/09/2010 SIN 23:59 - MXP 15:05	916€	Book!
British	08/09/2010	22/09/2010		

Figure 2-7. Conference Trip Planner Developed in MashArt

Analysis:

MashArt has shown that it is possible to develop a component-based application that can answer the example *conference search* problem, provided the necessary basic components are readily available. This highlights the potential of systems that provide casual users with consolidated access to multiple data sources. In terms of the output of the composition, it is interesting to note that while in the traditional search scenario the output is a set of result tuples, the output in mashArt is rather represented by the whole application, i.e., the individual components and their interconnection.

The mashArt project shows a promising approach to integrating Web 2.0 elements into mashups that support a casual user's querying of a specific domain. It provides a consolidated interface to multiple data sources of different types and provides guidance to users by pre-selecting useful components from a domain. Although the benefit of such applications from an end user's perspective is clear, mashArts' claims of supporting ordinary web users to compose their own mashups have yet to be experimentally validated. However its focus on minimising the costs involved in integrating separate components, and its emphasis on simplifying the composition process is to be welcomed.

2.2.8 Discussion

This section has detailed some of the varied ways in which ordinary web users can be supported in making complex queries over one or more knowledge bases. Table 2-1 summarises some of the features relating to these systems.

Table 2-1. Summary of Systems that Support Complex Querying by Casual Users

System	Multiple Sources	Multiple Data Types	Data Formats	SME	Main Query Interface[16]	Supports Application Development
Search Computing	Yes	Yes	Web Services and Relational DBs	Yes – no authoring tool	Widgets	Yes – Widget fronted search application
Parallax	No	No	Freebase Knowledge Base	No – auto generated facets	Facets	No
Sparallax	No	No	RDF	No – auto generated facets	Facets	Yes, but only by specifying an endpoint for their GUI
Explorator	Yes	No	RDF	No – auto generated facets	Facets & Query By Example	Yes, but only by specifying an endpoint for their GUI
PowerAqua	Yes	No	RDF	No	NLI	No
Orakel	No	Yes	OWL, F-Logic	Yes – has authoring tool (Framemapper)	NLI	Yes – can be integrated into bespoke application
MashArt	Yes	Yes	Web Services, Atom feeds, JSON, XML	Yes - has authoring tool (MashArt Editor)	Mashup Widgets	Yes - Mashup
SWP	Yes	Yes	RDF, Excel, Relational DB, Text files	Yes – has authoring tool (Ontology Management Component)	Mashup Widgets	Yes – Bespoke Portals

A number of important conclusions can be drawn from Table 2-1. The *Main Query Interface* column shows that there are a variety of ways to support users in forming complex queries and exploring datasets (faceted interfaces, natural language interfaces, widgets etc.), each of having their own strengths and weaknesses. Furthermore, some systems focus on question answering (e.g. PowerAqua and Orakel), whilst others support a more explorative approach to the data access (e.g. Parallax, Explorator etc.). As can be seen from the *Supports Application Development* column, some of the systems are standalone and do not enable applications being built on top of them. Furthermore, of those systems that did support application development, it is common for them to require a specific interaction paradigm (NLI, Widgets, etc.) to be used. This tightly couples any user interface with the underlying data sources. A more flexible approach would be to leave the

[16] Widgets refer to individual user interface items that users can interact with e.g. sliders that can be pushed, maps that can be zoomed into etc., Facets refer to faceted browsing interfaces, and Mashups refer to portals that contain multiple interacting widgets or applications.

choice of GUI to implement up to the application developers themselves. One way of achieving this is to provide an API that supports developers in creating a variety of user interface styles (basic query builders, browsers, sophisticated data visualisations etc.). The API should only involve the passing of parameters rather than full queries in a specific language (SPARQL, XQuery etc.), in order to minimise the learning curve for application developers.

Of those systems surveyed in Table 2-1 that support users through some manually generated Subject Matter Expertise (SME), all incorporated (or have plans to incorporate) a supporting authoring tool to simplify this process. There is a trade off between the benefits of offering manually generated expertise to users and the efficiency of providing automatically generated supports (e.g. dynamically created facets). Hence, the development of user friendly tools which are quick to use, but do not overly compromise in terms of the features they deliver (e.g. expressiveness of queries supported) should be greatly encouraged. As already described, Natural Language Interfaces like PowerAqua and ORAKEL have a similar dilemma, with the former opting for full automation and the latter providing a tool to support manual mappings by domain experts.

Of all the systems surveyed in Table 2-1, there are three which have manual SME authoring tools (Framemapper, MashArt Editor and the Ontology Management Component). Unfortunately whether any of the authoring tools can be successfully used by non-technical domain experts has not yet been experimentally validated. Since this research focuses on supporting users through manually generated SME, the development of an easy-to-use tool suitable for non-technical experts is a priority. Moreover this expertise should be encoded in a reusable model that can be plugged into different installations. This level of flexibility, and the reusability of SME, would make such a system more attractive to potential application developers. A state of the art survey of this area is presented in Section 2.3.

In terms of supported knowledge bases, Table 2-1 shows that applications vary from single repositories of a single data type (Parallax), to those that reconcile information over multiple resources of different data types (Search Computing). This reconciliation of data from multiple distributed sources is extremely powerful, and an attractive feature to include in the system design described in Chapter 3. One definite trend in this research area is the move to support RDF and especially Linked Data. Most systems have either been designed specifically with this format in mind (Sparallax), have added this functionality to the system (PowerAqua) or have plans to incorporate it in future releases (Search Computing). Again this has clear implications for the research described in this thesis and points towards the value of supporting queries over Linked Data. Furthermore, the use of dereferenceable URIs in Linked Data can help with the reconciliation of data from multiple sources, so this feature may also be useful to exploit.

With the proliferation of other data types beside RDF still widespread (XML, Web Service APIs etc.) it would also be prudent for systems to support multiple data formats like the Semantic Web Portal (SWP) does. While SWP approaches this by insisting that the sources must first be converted into a common data type (RDF), Search Computing allows different sources to coexist remotely in their own native format once registered to a service mart. This approach is more desirable as only the source schema needs to be registered beforehand, in contrast to the a priori transformation of an entire data set. Thus in this thesis it is vital that any such source registration model necessary to support integration is both lightweight and reusable in other installations, so that time constraints associated with this task are not overly arduous or off-putting for developers.

2.2.9 Key Findings

This section described a state of the art analysis of *Complex Querying by Casual Users over Multiple Sources*. A number of related systems have been used as case studies and reviewed, with the important features of each summarised in a comparative table (Table 2-1). A discussion of these systems was then conducted. From the analysis of this area, the following points have been identified as critical influences on the design of the system outlined in this thesis:

- There is a lack of support for third party applications who wish to incorporate complex querying mechanisms into their design, whilst maintaining the freedom to choose what interaction paradigm (facets, widgets, mashups and NLI etc.) their users may follow. This often restricts applications to offering either browsing or question answering as a way to access the data.
- In terms of supporting the reconciliation of data from multiple heterogeneous sources, the best practice in terms of scalability and integration cost is to allow each information source to co-exist in their own native format. Furthermore, Linked Data is an increasingly popular methodology which should be supported.

- Though manually generated SME has been successfully deployed in bespoke expert systems, in general purpose systems that support complex querying over multiple sources it has not been widely utilised (with the exception of some NLI and mashup systems). There is considerable potential in this area for such functionality to be exploited, as users often welcome quality guidance in domains unfamiliar to them or those they would like to learn more about.

2.3 Domain Independent Tools to Support SME Encoding by Non-Technical Experts

It has been highlighted how SME can play a crucial role in mechanisms that support casual users perform data exploration. The issue of the manual encoding of SME by non-technical users is now considered in more detail by describing a state of the art analysis of *Domain Independent Tools to Support SME Encoding by Non-Technical Experts*. It should be reiterated that this thesis takes a broad view of how SME can be encoded, as it discusses multiple ways to approach the encoding of domain expertise. For instance, SME can be encoded as an ontology, as a rule in a query language, or as a mashup (someone using their "expertise" to combine data sources that are beneficial to ordinary data users). All these methods offer different features, and it is possible to encode the same SME in a number of different formats. In particular this section concentrates on examining systems that can be repurposed to different domains and that aim to support non-technical users[17] in encoding SME.

This section contains a number of case studies of such SME encoding systems. The systems chosen typically have a high impact value within the literature or follow an approach that is related in some way to that which is adopted by this thesis. The set of systems included also aims to reflect the diversity in the literature in terms of approaches to SME encoding. These systems include query building software, mashup composition tools, as well as ontology construction tools. In the course of analysing SME encoding systems for Research Objective 1 (Section 1.3), the following criteria with which to evaluate these systems were derived:

1. How does it encode SME?

[17] As mentioned previously, this thesis defines non-technical experts as people that have basic computing skills such as Internet browsing, but no formal background in computer programming or data modelling.

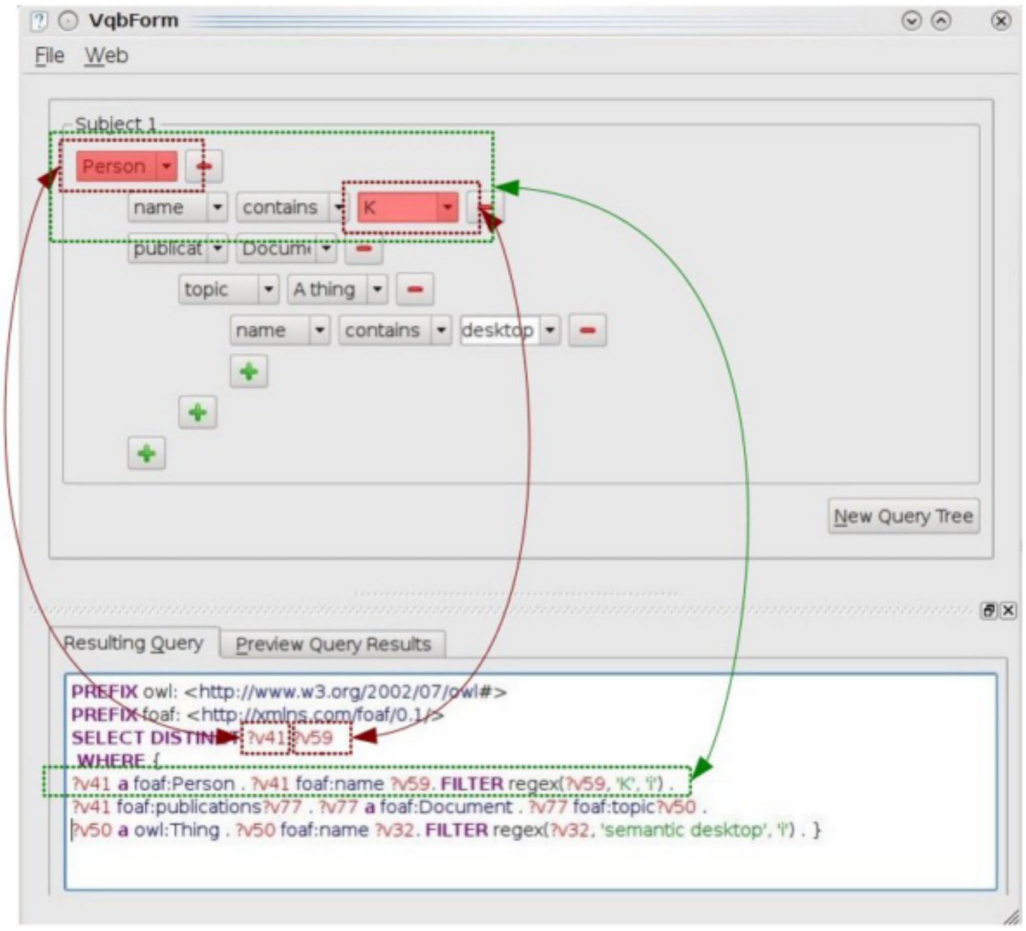
2. Has the system been successfully evaluated with non-technical users?
3. Does the system work with multiple knowledge bases?
4. Does the system work with knowledge bases of different data types?
5. Can the SME be tailored by end users to their own interpretation?
6. Can the SME be reused as it is in other installations?

2.3.1 Konduit VQB

Description:

Konduit VQB (Möller et al.) is a visual query builder that aims at assisting users in building queries to search over RDF data stored on the Nepomuk Social Semantic Desktop. Given that the amount of semantic data on an individual's desktop is constantly growing, there is a need to support these users in accessing this data effectively. One way to tackle this problem is to use Konduit VQB to support users in visually creating SPARQL queries to search the RDF data. Interestingly, the tool is aimed at users with little or no knowledge about SPARQL, as well as those users who are more familiar with Semantic Web technologies. Furthermore the tool also supports queries over a repository of multiple ontologies, as opposed to other tools which restrict querying to a single ontology. Though Konduit VQB is not strictly an SME encoding tool, it could be used by non-technical experts to encode SME in SPARQL, with this SME then leveraged by users wishing to explore an RDF repository. Hence it is this aspect of the tool which is of most relevance to this thesis that is analysed.

Employing SPARQL as an SME encoding format implies a reliance on the expert to generate queries (rules) based on the metadata schema, in order to return specific instances. Hence this section will restrict its discussion of Konduit VQB to its schema-based approach to generating SELECT queries, as this is of most relevance to this thesis. Interestingly, the developers of Konduit VQB feel that the schema-based SELECT query builder in the tool incorporates their most user-friendly approach to building SPARQL queries, and recommend it for those users having the least knowledge of semantic technologies. This is because the approach requires less input from the user, and as such it is particularly relevant to non-technical domain experts wishing to encode SME as a SPARQL SELECT statement.



```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?v41 ?v59
 WHERE {
?v41 a foaf:Person . ?v41 foaf:name ?v59. FILTER regex(?v59, 'K', 'i') .
?v41 foaf:publications?v77 . ?v77 a foaf:Document . ?v77 foaf:topic?v50 .
?v50 a owl:Thing . ?v50 foaf:name ?v32. FILTER regex(?v32, 'semantic desktop', 'i') . }
```

Figure 2-8. Konduit VQB Interface for Creating SELECT Statements in SPARQL

Analysis:

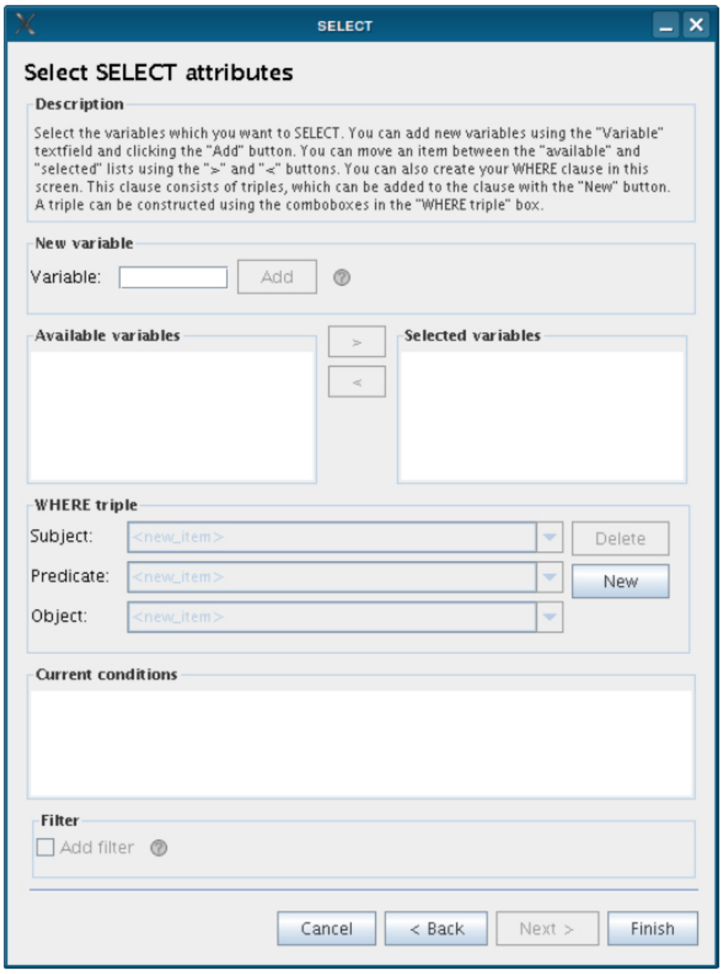
Konduit VQB assumes that users most often want to find certain information on entities that have restricting characteristics e.g. one might want to search for a person (entity) with a given name (restricting characteristic). The interface therefore provides a way to build queries as trees, starting with the type of entity the user is looking for (e.g. person, document) and progressing with restrictions (e.g. name, size) on the branches of the tree. Hence a user initially selects the entity they want to find, and then adds any available restrictions (selected from a dropdown box populated by properties from the schema) to the tree underneath. If a property has a literal range, the user can enter a value and restrict it on a relation, such as *equals* or *contains*. According to the developers, the advantage of this approach is that it is simple, intuitive and satisfies the large number of occasions when the user wants to search for something based on certain properties (Ambrus et al. 2010). This particular approach however does not support the full range of expressivity that SPARQL offers. The developers feel that the extra work involved in forming more complex queries (e.g. using the CONSTRUCT query builder they offer) is likely to make them inaccessible to users with no knowledge of RDF and SPARQL.

The developers have plans to perform usability evaluations to determine the most appropriate methods to support query building by complete novices, as well as those with deep knowledge of semantic technologies. Until these experiment results are presented, it will not be conclusively demonstrated that casual users can use the tool effectively. However it is clear that out of the four approaches that the tool offers, the schema-based approach to generating SELECT statements is the most suited to non-technical users.

2.3.2 SPARQLViz

Description:

SPARQLViz (Borsje & Embregts 2006) contains a Graphical Query Composer that allows users to generate syntactically correct SPARQL queries through a wizard interface. It is particularly useful for novice users who have little or no experience with SPARQL, as the user is able to compose a valid query simply by using familiar user interface widgets in a wizard-like manner (Borsje & Embregts 2006). A drawback of solely creating SPARQL queries using a wizard is that the user can lose some flexibility in terms of expressiveness, as they are constrained to the specific template defined in the GUI. This is a common issue with query building applications, where a balance needs to be struck between faithfulness to the query language's expressivity and the ease of use of its GUI metaphor. SPARQLViz is very faithful to the SPARQL language as it supports all four types of its queries (SELECT, CONSTRUCT, DESCRIBE and ASK), and implements major features like PREFIX, DISTINCT, ORDER BY, LIMIT and OFFSET. However, due to varying levels of complexity, not all these query types and features may be usable by novice users of the tool.


Select SELECT attributes

Select the variables which you want to SELECT. You can add new variables using the "Variable" textfield and clicking the "Add" button. You can move an item between the "available" and "selected" lists using the ">" and "<" buttons. You can also create your WHERE clause in this screen. This clause consists of triples, which can be added to the clause with the "New" button. A triple can be constructed using the comboboxes in the "WHERE triple" box.

Figure 2-9. SPARQLViz Interface for Creating SELECT Statements in SPARQL

The SPARQLViz GUI has a number of features aimed at making it as user-friendly as possible. For instance an emphasis has been placed on standardisation, meaning that every screen is constructed in the same way i.e. contains a title and a description block, with all other functions logically grouped within their own block. Where needed, there is a question mark icon which displays help information as a tooltip, and at the end of each screen there are four buttons which provide the user with means to navigate through the wizard. To further support the non-technical user, SPARQLViz employs many common widgets which a user would likely be familiar with from other programs or Internet forms (e.g. radio buttons, checkboxes, listboxes, lists, text fields and buttons). Another feature geared towards simplifying the query generation process is the use of semantic dependencies within the RDF to limit the choices that a user can make when creating conditions. For instance, if the user chooses a specific *subject*, the number of related *predicates* which can be validly selected is displayed. Likewise if the user selects a specific *predicate*, only the *objects* to which that *predicate* is connected to are displayed as viable choices.

Analysis:

According to the developers, the key features of the approach offered by SPARQLViz are the fact that generated queries will always be valid which saves a lot of debugging time, and also that an in-depth knowledge of the SPARQL query language is no longer necessary to generate them. Unfortunately there have been no experiments conducted regarding the second assertion, so it is not possible to determine what level of SPARQL expertise (if any) SPARQLViz requires of its users in order to generate queries. The fact that users have to manually add variables to the system in order to generate basic queries makes it unlikely that a casual user would be able to use the system without some understanding of SPARQL. One further limitations of the system is that it only allows queries to be formed over a single RDF data source, however it overcomes this somewhat by generating native SPARQL, so that the outputted queries can be reused in any application that contains the same RDF. Overall SPARQLViz shows that its wizard approach to query formulation, if it is simplified to the correct degree, has the potential to be used by non-technical users in order to generate expert rules in SPARQL.

2.3.3 Potluck

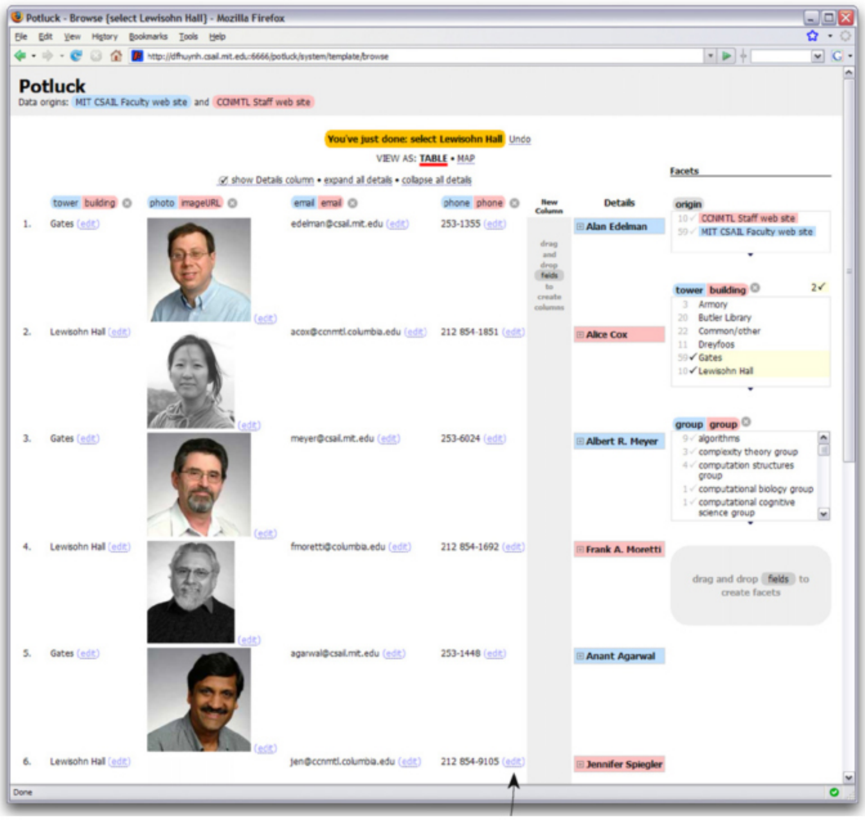
Description:

Potluck is a web user interface that aims at allowing casual users (those without programming skills and data modelling expertise) mash up data themselves (Huynh et al. 2008). Hence Potluck can allow non-technical domain experts to create applications that encompass their SME, in order to support users who are interested in that domain in finding useful information from multiple sources. One major limitation of Potluck is that it only enables the mashing up of web pages that use the Exhibit[18] framework. Though Exhibit is useful software in its own right and allows website authors to create dynamic exhibits of their collections without resorting to complex database and server-side technologies, this is quite a severe restriction. The main features of Potluck are that it:

[18] http:// simile-widgets.org/exhibit

- allows the user to merge fields from different Exhibit data sources, so that they are treated identically for sorting, filtering, and visualisation. Fields are merged using simple drag and drop of field names.
- provides an efficient means for the user to clean up data syntactically, homogenise data formats, and extract fields syntactically embedded within existing fields, all through the application of simultaneous editing.
- supports faceted browsing to let users explore and identify subsets of data of interest or subsets of data that need alignment and cleaning up (Huynh et al. 2008).

By simply dropping a field tag onto an existing column or facet, a merged field is generated that contains data from both sources e.g. dropping the "photo" field onto the existing "imageURL" column creates a merged "photo/imageURL" column. The *edit link* next to each field value opens up the *simultaneous editing* dialog box where the values of that particular field can be edited en masse. This is useful for correcting inconsistencies between data sets that occur many times, such as prefixing area codes to phone numbers or wrapping existing area codes in parentheses. It is also useful for reformatting a field, such as changing "first-name last-name" into "last-name, first-name". Potluck also provides two ways of visualizing data, a tabular view and a map view.


Potluck - Browse [select Lewisohn Hall] - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dfhuynh.csail.mit.edu:6666/potluck/system/template/browse
Potluck
Data origins: MIT CSAIL Faculty web site and CCNMTL Staff web site
You've just done: select Lewisohn Hall Undo
VIEW AS: TABLE • MAP
show Details column • expand all details • collapse all details
New Column
drag and drop fields to create columns
Facets
drag and drop fields to create facets
Done

	tower building	photo imageURL	email email	phone phone	Details
1.	Gates (edit)	(edit)	edelman@csail.mit.edu (edit)	253-1355 (edit)	Alan Edelman
2.	Lewisohn Hall (edit)	(edit)	acox@ccnmtl.columbia.edu (edit)	212 854-1851 (edit)	Alice Cox
3.	Gates (edit)	(edit)	meyer@csail.mit.edu (edit)	253-6024 (edit)	Albert R. Meyer
4.	Lewisohn Hall (edit)	(edit)	fmoretti@columbia.edu (edit)	212 854-1692 (edit)	Frank A. Moretti
5.	Gates (edit)	(edit)	agarwal@csail.mit.edu (edit)	253-1448 (edit)	Anant Agarwal
6.	Lewisohn Hall (edit)		jen@ccnmtl.columbia.edu (edit)	212 854-9105 (edit)	Jennifer Spiegler

origin
10 ✓ CCNMTL Staff web site
59 ✓ MIT CSAIL Faculty web site

tower building 2✓
3 Armory
20 Butler Library
22 Common/other
11 Dreyfoos
59 ✓ Gates
10 ✓ Lewisohn Hall

group group
9 ✓ algorithms
3 ✓ complexity theory group
4 ✓ computation structures group
1 ✓ computational biology group
1 ✓ computational cognitive science group

Figure 2-10. Screenshot of Potluck showing several columns and facets of merged fields.

Analysis:

A user study was conducted to ascertain whether people could learn how to use Potluck, as well as to discover usability problems. There were eleven subjects (five librarians and six general users) involved in the trial, and in general subjects successfully used the drag and drop metaphor to merge fields and create facets. There was no noticeable difference between the subjects from the general population and the librarians who work with metadata on a daily basis, however the programmers (one from the general population and one librarian) appreciated the functionality of Potluck more.

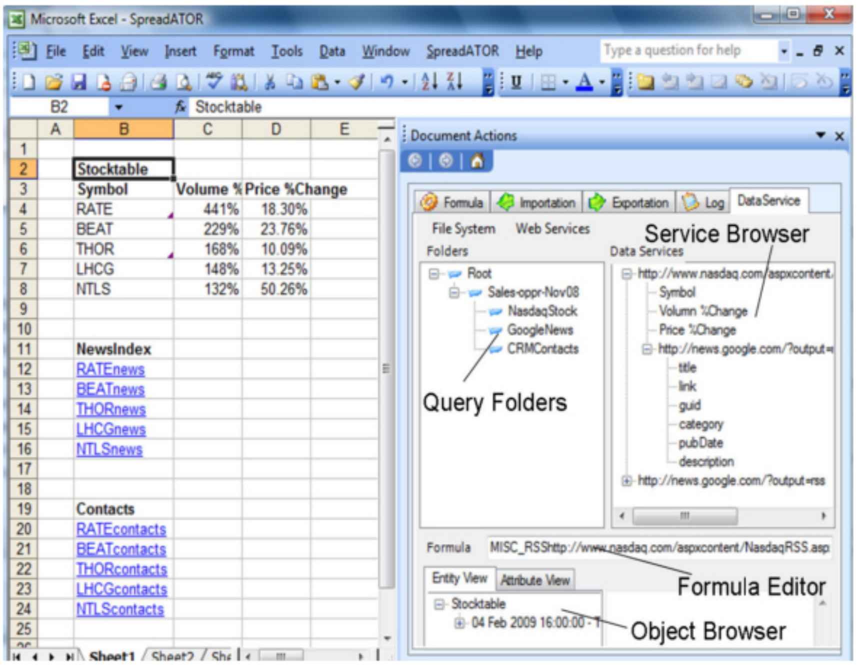
Potluck shows that it is possible for casual users to generate mashup applications, and the developers feel that in the future when more reusable data becomes available, interfaces like that of Potluck have the potential to level the playing field for non-programmers. Expanding Potluck's compatibility beyond the scope of Web pages embedded with Exhibit would help this situation enormously, as this constraint considerably narrows the potential data available to it. Furthermore, experiments preformed with Potluck were of a very small size (data sets with hundreds of entities) so there may be issues with scalability. However, overall Potluck does show potential as a mechanism for non-technical domain experts to encode their SME as a mashup.

2.3.4 SpreadATOR

Description:

Current APIs are designed for developers with programming expertise, and thus are not directly usable by a wider class of users who do not have a programming background, but would nevertheless like to build their own mashups. To address this need, a spreadsheet-based Web mashup development framework has been proposed (Kongdenfha et al. 2009), which enables users to develop mashups in the popular spreadsheet environment. A system based on this framework called SpreadATOR (Saint-Paul et al. 2008), makes it possible to access a variety of Web data sources, represent them on a spreadsheet, and manipulate the data (including imposing changes on the data source) from the "comfort" of the spreadsheet.

An example application created in SpreadATOR was a *sales opportunity identification mashup*, in which data was aggregated and combined from three different data sources: Nasdaq RSS service, Google RSS News service, and a Customer Relationship Management (CRM) system. This mashup monitors stock markets, looking for companies with the largest gains in their stock prices. A strong rise of a company's stock is often a sign that a significant event just happened in the company, and any such event may be an opportunity for selling software to the company. Thus a salesperson using this application can see the five stocks with the biggest gains from Nasdaq.com and then get more information about each stock's company in the list (news related to each stock from Google news service, and contact details and purchase histories of each stock's company from the CRM system).



	A	B	C	D	E
1					
2		Stocktable			
3		Symbol	Volume %	Price %Change	
4		RATE	441%	18.30%	
5		BEAT	229%	23.76%	
6		THOR	168%	10.09%	
7		LHCG	148%	13.25%	
8		NTLS	132%	50.26%	
9					
10					
11		NewsIndex			
12		RATEnews			
13		BEATnews			
14		THORnews			
15		LHCGnews			
16		NTLSnews			
17					
18					
19		Contacts			
20		RATEcontacts			
21		BEATcontacts			
22		THORcontacts			
23		LHCGcontacts			
24		NTLScontacts			
25					

Formula: MISC_RSShttp://www.nasdaq.com/aspxcontent/NasdaqRSS.asp

Figure 2-11. SpreadATOR User Interface

SpreadATOR (Kongdenfha et al. 2009) aims at supporting users proficient in spreadsheets, but not computer programming, to develop such a mashup through the following five mechanisms:

- It provides a tool to hide the heterogeneity of the source data access methods (e.g. HTTP or SQL queries) and the source data representations (e.g., XML or JSON). Typically, adapters are required to map data formats, and to manipulate operations between SpreadATOR and underlying data sources.
- It has a tool to cater for both simple query specification (e.g. to filter unwanted data), and presentation (e.g. to display large sets of data on the spreadsheet) as the data obtained from the CRM system may contain thousands of records.
- It supports the presentation of stock data and news data in different ways e.g. stocks in tabular form and news stories as a list of hyperlinks.

- It allows manipulation of details on the spreadsheet that can push the changed data back to the CRM system after manipulations.
- It provides users with ability to browse up-to-date information on the spreadsheet as stock data and news are frequently updated.

Analysis:

SpreadATOR is another example of a system aiming to support the development of bespoke mashups by users without a computer programming background (though admittedly any user would need to have considerable experience with spreadsheet applications). Such a tool, if embraced by end-users, would enable experts to encode their SME as a complete mashup application. SpreadATOR's support for the reuse and customisation of components could potentially reduce the time necessary to create mashups, making it more attractive to users. This is important, as any mashup should ensure that the benefits of using it outweigh the effort involved in its generation. Finally by allowing the integration of multiple data types into a single mashup (though wrappers typically need to be generated), SpreadATOR enables a wide variety of data to be exploited from the domain expertise of the mashup creator. Considering the heterogeneity of data sources present on both the Web and within individual organisations, this is an important feature.

Unfortunately SpreadATOR is still in prototype stage and has not yet been evaluated with real end users. This means that the developers' claims that their experiences "demonstrate the superiority of their spreadsheet-based mashup tool compared to existing tools in terms of both simplicity and development productivity" (Kongdenfha et al. 2009) cannot be validated. Until experiments are conducted that show users accept the spreadsheet paradigm for creating mashups and that the framework supports more efficient mashup development, then the true potential of SpreadATOR will not be known.

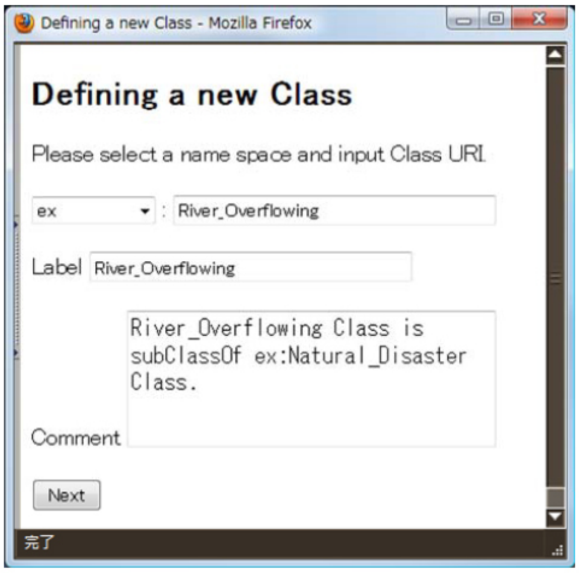
2.3.5 Web Ontology Building System for Novice Users: A Step-by-Step Approach

Description:

Web ontologies play a central role in Semantic Web applications, and are utilised for various purposes. However, ontologies are not widely exploited in many applications, in part because ontology construction is time consuming and requires expert knowledge (Yasunaga et al. 2010). There are many approaches to building ontologies, such as automated ontology construction using machine learning techniques (Maedche & Staab 2001), and ontology editing tools to facilitate ontology development by human users (Kozaki & Mizoguchi 2005). However while an ontology editor is an indispensable tool for building ontologies, the use of such an editor generally requires extensive knowledge of ontologies to begin with, and is therefore not suitable for novices (Yasunaga et al. 2010). Allowing novice users to create, and collaborate on, ontologies which encode their SME would likely increase the adoption of ontology editors. Hence a Web ontology building system especially targeted for novice users has been proposed by the team in Ritsumeikan University (Yasunaga et al. 2010). It contains the following features:

- The process of building an ontology is broken into small and simple steps, and the proposed system provides users with step-by-step instructions.
- The proposed system restricts each step in the building process to a simple task such as entering the information into the form or choosing a word from a list of vocabularies.
- The ontology file is converted into an RDF graph and presented visually to users.
- By implementing as a web application, users can use a web browser to build an ontology. Thus, distributed and cooperative development can easily be achieved.
- The proposed system provides a checking function that infers data with built-in OWL definitions.

To use the Ritsumeikan University tool, a user first makes a decision about the domain covered by the ontology and names a corresponding project e.g. "Disaster_Mitigation_Ontology". If the user needs to reuse an existing ontology, he can upload an ontology file. The next step is to define a new class by choosing a prefix "ex:" and entering the URI for each class in text form e.g. "ex:River_Overflowing" or "ex:Natural_Disaster". In addition, users can enter labels or comments. Next a semantic hierarchy of the classes is defined e.g. the "River_Overflowing" class is a subclass of the "Natural_Disaster" class. Defining properties is similar to the process of defining the classes themselves. When users finish defining the ontology, the system displays it to the user graphically. Finally, the system provides the user with a tool to check what data can be inferred from the ontology. In this stage, the user uploads an RDF file that he wants to check, and new data is derived from this file by the ontology they have just created. Users can then view this inferred data and confirm whether the correct results are being inferenced by the ontology.


Defining a new Class - Mozilla Firefox

Defining a new Class

Please select a name space and input Class URI

ex : River_Overflowing

Label River_Overflowing

Comment River_Overflowing Class is subClassOf ex:Natural_Disaster Class.

Figure 2-12. Screenshot of a New Class Being Defined in the Ritsumeikan University Tool

Analysis:

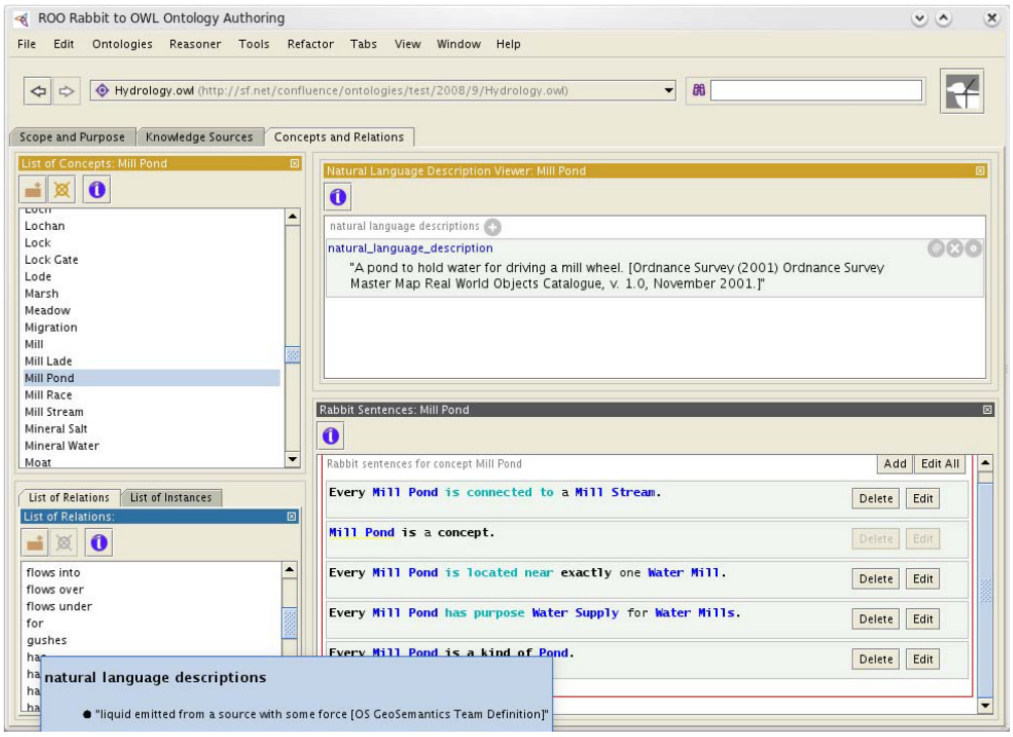
The step-by-step ontology building approach aimed at novice users is a potentially useful way of popularising ontology creation for the Semantic Web. However, the tool has yet to be evaluated with any users (experienced ontology creators or novices) so its efficacy has yet to be validated. Encoding SME in this manner could contribute to the wider use of Semantic Web technology systems. However it is still unclear whether this system is any easier to use than traditional ontology editing tools, such as Protégé[19], or whether it supports novice users in encoding useful SME without specific training or help from knowledge engineers. Furthermore, even after a useful ontology is encoded, there is likely to be a further integration or mapping process necessary in order to deploy it over a knowledge base populated with instances. This would likely require some input from someone knowledgeable in semantic interoperability, which would be beyond the capabilities of a novice ontology creator. Therefore, a well generated ontology from this tool might not be able to be immediately deployed in an application, which slightly narrows its appeal as a way of encoding SME.

[19] http://protege.stanford.edu/

2.3.6 ROO

Description:

Recent work on ontology engineering has seen the adoption of Controlled Natural Languages (CNL) in some systems to ease the process of ontology authoring. CNLs are subsets of natural languages, which reduce ambiguity by restricting vocabulary and grammar. However to be efficient, CNL-based tools still require good knowledge engineering skills. As a consequence, an ontology authoring tool called ROO (Denaux et al. 2010), has been developed to cater for the needs of domain experts with little or no ontology engineering experience. ROO is based on the Protégé ontology editor and assists domain experts to build conceptual ontologies using a CNL-based interface. The CNL that ROO uses is called Rabbit (Dolbear et al. 2007).


ROO Rabbit to OWL Ontology Authoring
File Edit Ontologies Reasoner Tools Refactor Tabs View Window Help
Hydrology.owl (http://sf.net/confluence/ontologies/test/2008/9/Hydrology.owl)
Scope and Purpose
Knowledge Sources
Concepts and Relations
List of Concepts: Mill Pond
Lochan
Lock
Lock Gate
Lode
Marsh
Meadow
Migration
Mill
Mill Lade
Mill Pond
Mill Race
Mill Stream
Mineral Salt
Mineral Water
Moat
List of Relations
List of Instances
List of Relations:
flows into
flows over
flows under
for
gushes
Natural Language Description Viewer: Mill Pond
natural language descriptions
natural_language_description
Rabbit Sentences: Mill Pond
Rabbit sentences for concept Mill Pond
Add
Edit All
Delete
Edit

"A pond to hold water for driving a mill wheel. [Ordnance Survey (2001) Ordnance Survey Master Map Real World Objects Catalogue, v. 1.0, November 2001.]"

Every Mill Pond is connected to a Mill Stream.

Mill Pond is a concept.

Every Mill Pond is located near exactly one Water Mill.

Every Mill Pond has purpose Water Supply for Water Mills.

Every Mill Pond is a kind of Pond.

natural language descriptions

- "liquid emitted from a source with some force [OS GeoSemantics Team Definition]"

Figure 2-13. Screenshot of the ROO Application

ROO automatically translates Rabbit sentences generated by the domain expert into OWL, and the main features of ROO (Denaux et al. 2010) for novice users who wish to generate ontologies are that it:

- avoids using technical terminology, preferring to use conceptual terminology which is easier to understand for novice users.
- helps users to avoid introducing ambiguity in the resulting ontology. It avoids making assumptions by requiring the declaration of concepts, relationships and

individuals. ROO is also aware of cases when parsed sentences could be ambiguous and warns the user accordingly, preferably suggesting ways to avoid ambiguity.

- provides guidance about how to build ontologies that are easy to reuse. ROO incorporates a model of Kanga (Kovacs et al. 2006), an Ontology Engineering methodology, and can make suggestions to the user regarding tasks that need to be performed to improve the reusability and general quality of the ontology.

Analysis:

An evaluation of ROO was conducted with 16 students (unfamiliar with ontology encoding) from university departments of geography and earth & environment. During this study participants were asked to perform ontology modelling tasks similar to those that would be expected of domain experts by the Ordnance Survey. The study showed that the students were able to perform ontology authoring tasks after only receiving a training of about 10 minutes. Regarding usability, ROO was rated as intuitive to use and the students did not find the error messages confusing. In fact they felt that the messages helped them to write correct sentences.

Unfortunately an analysis of the quality of the resulting ontologies showed that they were incomplete and not fit for purpose. Furthermore, modelling problems within the ontologies also occurred, such as multiple tangled inheritances. The study did show that the use of a CNL interface to build ontologies makes it possible for domain experts to start creating ontologies, but was not enough to avoid the occurrence of modelling errors which result in user frustration. It was also felt that the use of further intelligent tool support, which caters specifically for novice users, is essential to improve the usability of tools like ROO. Despite the benefits of CNLs, referring to CNL vocabulary/syntax rules can be time consuming, annoying and in certain cases may prevent uptake of the tool (Davis et al. 2010). Furthermore, ROO does not support a full roundtrip process, where an ontology can be parsed into Rabbit sentences, edited as required, with these sentences subsequently generated back into an ontology.

ROO has shown potential in supporting domain experts with no ontology engineering experience in creating useful ontologies, although it is still not sufficiently intuitive a tool to support this process without the intervention of a knowledge engineer. Moreover, the innate complexity and richness of OWL may be a serious challenge for users without prior training in ontology construction to generate fully formed and useful ontologies. As stated earlier, even when a useful ontology is encoded, there is likely to be a further integration or mapping process necessary to deploy it over a knowledge base populated with instances. This would be beyond the capabilities of a novice ontology designer, and would typically involve a person familiar with the issues of semantic interoperability. Though not a problem per se, it may limit the immediate applicability of the ontologies generated by the tool.

2.3.7 Discussion

This section has detailed some of the varied ways in which non-technical domain experts can encode SME in domain independent tools. Table 2-2 summarises some of the features relating to these systems.

Table 2-2. Summary of Domain Independent Tools that Support Non-Technical Domain Experts in Encoding SME

Software	**SME Generated**	**Multiple Data Types**	**Tailorable SME by end user**	**Evaluated by Non-Technical Users**	**Reusable**	**Multiple Sources**
Konduit VQB	SPARQL queries	No	No	No	Yes	Yes
SPARQLViz	SPARQL queries	No	No	No	Yes	No
Potluck	Mashup up of Exhibit based sources	No	No	Yes	No	Yes
SpreadATOR	Spreadsheet based Mashup	Yes – XML, JSON, Relational DB	No	No	No	Yes
Web Ontology Building System for Novice Users	RDFS / OWL	Yes – OWL / RDFS	No	No	Yes	N/A
ROO	OWL	No	No	Yes	Yes	N/A

The *SME Generated* column of Table 2-2 highlights some of the varied ways in which non-technical domain experts can encode SME such as SPARQL queries, mashups and OWL. There are strengths and weaknesses to each system's approach which have been pointed out in the relevant case studies. In terms of using a visual query builder to encode SME, a form or wizard based approach (Konduit VQB and SPARQLViz) appears to offer the most potential for non-technical domain experts due to their relative simplicity. Specifically, the more intuitive schema-based approach for generating statements was mentioned as the most suited to non-technical users. An example of a commercial application that uses such a rule building approach is iTunes[20], which has a *smart playlist*[21] builder. The *smart playlist* builder is aimed at ordinary end users and allows them to generate XQuerys in a schema-based approach. These XQuerys generate specific music playlists that allows users to better manage and sort their music collection.

At smartplaylists.com[22], users can share the rules they have encoded in iTunes, so that others can benefit from their expertise. This supports the notion that SME encoded in this way can be a valuable way of supporting other users to explore and manage their day to day data. The smart playlist feature in iTunes has even been used to organise PDFs about radiology (Qian et al. 2008) which further supports the contention that a schema-based approach to rule generation, coupled with a form/wizard interface is accessible to a non-technical user of computers. There are other approaches to visually building queries such as the use of graphs which are employed by NITELIGHT (Russell et al. 2008) and iSPARQL[23]. However, in order to use these tools the user must have a full comprehension of the underlying RDF schema and the query language syntax, which implies a high cognitive load for newcomers and less experienced users (De Araújo & Schwabe 2009). Indeed the developers of NITELIGHT admit that the close correspondence between the graphical notations and query language constructs makes the tool largely unsuitable for users who have no previous experience with SPARQL. Hence, the form/wizard interface appears to be more suited to non-technical domain experts, however this remains to be shown experimentally.

Using an ontology as a means of encoding SME is a common approach employed by domain experts. However, the use of an ontology editor such as Protégé to develop usable ontologies often requires specialist skills in ontology engineering (Davis et al. 2010). Professionals (clinicians, business analysts, legal experts etc.) should not be expected to upskill themselves to comprehend Semantic Web formalisms, and the process of knowledge gathering involving both domain expert and knowledge engineer can be time-consuming and costly (Davis et al. 2010). The step-by-step ontology building approach for novice users has been proposed as a solution to this problem, because it caters for domain experts with no experience of ontologies. However, as seen in Table 2-2 the tool has yet to

[20] http://www.itunes.com
[21] http://support.apple.com/kb/HT1801
[22] http://www.smartplaylists.com/
[23] http://demo.openlinksw.com/isparql/

be evaluated with any users so its efficacy has not been validated. ROO is an alternative solution that uses a CNL interface to support novice ontology builders. Unfortunately, despite some very useful features, the systems evaluation showed that completed ontologies were not fit for purpose, with problems such as multiple tangled inheritances. Furthermore, one of the main benefits of an OWL ontology is that it can be processed and interpreted by algorithms. Unfortunately, this typically means that further intervention by computer programmers is required before the benefits are felt by casual users who need support in exploring a specific domain of information.

Experts can also use tools that support the generation of mashups to encode their SME. End users could benefit massively from the guided support associated with a mashup when exploring heterogeneous sources in a consolidated fashion. However generating these mashups is non-trivial for people without computer programming experience, and accordingly many tools are being developed for this purpose. SpreadATOR is one such tool that uses the spreadsheet paradigm familiar to many computer users. The spreadsheet approach may be promising; however its lack of any user evaluation means its potential remains unclear. Potluck did have some degree of success with a small user trial involving non computer programmers; however it has limitations regarding the scale and scope of data sources it can accommodate. This is very problematic in terms of supporting access to large scale heterogeneous sources.

In a wider context, wire or pipe based tools have been presented as mechanisms that ordinary consumers can use to generate their own mashup applications. However as described below, many are not convinced that these tools are intuitive enough for non-technical domain experts to use without the investment of appropriate training. For instance it is felt that the level of abstraction of Yahoo! Pipes'[24] operations, and the characteristic data flow logic is only barely understandable to non-programmers (Daniel et al. 2010). Others suggest that it is too difficult for both casual users and power users to create personalized mashup applications in an appropriate time (Fischer et al. 2009). This is echoed by Wong and Hong who argue that most tools still require too much familiarity with web technologies and programming, and focus mainly on lightweight user interfaces (Wong & Hong 2007). The main limitations of these tools appear to be a lack of any mechanism to select data sources efficiently, and the time consuming nature of the manual

[24] http://pipes.yahoo.com/pipes/

composition of different components. (Fischer et al. 2009). The length of time to compose mashups also increases with the number of components available, and rapidly makes mashups more complex. Hence this thesis will focus on providing a mechanism for non-technical domain experts to generate SME via a set of individual rules. These rules will work over data sources rather than burdening the domain expert with the task of integrating heterogeneous data sources and a GUI into a unified application. These processes are better tackled by developers since no appropriate tools to support non-technical users are currently available.

This section has examined three ways of encoding SME (visual query builders, ontology builders and mashup constructors). It is argued in this thesis that the approach with the most potential to be adopted by non-technical domain experts is that of visual query building due to its relative simplicity. There may be restrictions on the potential richness of the SME that can be ultimately encoded; however a well designed system with access to rich metadata should provide sufficient scope for domain experts to create useful rules. Moreover, by supporting the combination of multiple rules into compound queries, relatively simple queries can play a part in more complex requests. This thesis also adopts a schema-based approach in a wizard/form interface as a way of supporting novice users in this paradigm, because it appears to be an intuitive approach widely used in query building applications. An evaluation of such an approach with non-technical users will be necessary to validate this decision.

While visual query builders typically allow queries to be sent to a data set, this thesis proposes to encapsulate these queries as part of an SME model. This allows the SME encoded to persist as part of a reusable model, rather than transiently exist as part of a query. Furthermore, since query languages support the integration of variables within queries, it should be possible for the SME to support tailoring of rules by end users who submit parameters. This would allow the semantics of a rule (and the range of instances returned by it) to be adjusted to the end user's perspective, and would be particularly useful for expert rules of a more subjective nature. As can be seen in Table 2-2 however, none of the systems analysed offered this functionality. A common issue with visual query builders is that they typically only support a single query language. This is very limiting given the variety of data formats that exist in common usage. Hence any tool using queries to encode SME should support the generation of multiple query formats (in as uniform a manner as possible), as well as access to multiple data sources.

2.3.8 Key Findings

This section examined the state of the art analysis of *Domain Independent Tools to Support SME Encoding by Non-Technical Experts*. A number of related systems have been used as case studies and reviewed, with the important features of each summarised in a comparative table (Table 2-2). A discussion of these systems was then conducted. From the analysis of this area, a number of important conclusions have been identified that have implications for the chosen system design described in Chapter 3:

- Overall there is a lack of domain independent tools that support non-technical domain experts to encode useful SME.
- An SME encoding approach that offers significant potential of being adopted by non-technical domain experts is visual query building. Specifically a schema-based approach in a wizard/form interface appears to be the most intuitive way of supporting novice users.
- The best practice is for SME to be reusable in different installations, so any queries created by domain experts should be encapsulated as rules in a reusable SME model.
- There is a general lack of support for users who wish to tweak or tailor the SME encoded in these tools beyond inputting a keyword to focus a search.
- Many SME encoding tools only support a single data format and many only operate over a single source.
- Many tools purporting to be designed for use by non-technical users have not yet been evaluated with such users.

2.4 Conclusion

This chapter focused on a state of the art analysis of *Complex Querying by Casual Users over Multiple Sources* and *Domain Independent Tools to Support SME (Subject Matter Expertise) Encoding by Non-Technical Experts*. The main approaches were examined critically, with advantages and disadvantages of each system identified. Key finding from this analysis were summarised in Section 2.2.9 and Section 2.3.8, and are used to directly inform the system design described in Chapter 3.

3 Design

This chapter describes the design of a framework and approach that fulfils the goals and objectives outlined in Section 1.3. The framework's design is also influenced by Chapter 2's analysis of the state of the art, and consists of design time and run time components. These components, and the various models and interfaces necessary to support this novel framework, are also discussed in detail. The chapter concludes with a description of the *expert-supported approach to data exploration*, which defines the end-to-end processes that underpin the framework and models.

3.1 Introduction

From the analysis of the state of the art in Chapter 2, it is clear that there is still a compelling need for frameworks that support casual users in exploring and querying multiple data sources from a domain. Furthermore, there is a general lack of domain independent tools to support the encoding of Subject Matter Expertise (SME) by non-technical experts. This chapter first describes how the state of the art has influenced the design of a framework which helps casual users perform data exploration. In the context of this thesis "data exploration" is defined as supporting complex queries against structured or semi-structured data, in order to locate important and relevant data for further analysis. The central role of the human in data exploration is a key differentiating factor between it and data mining, with data mining predominantly using algorithms to automatically search for patterns and features in large data sets.

After discussing the design influences stemming from the state of the art, the chapter continues with a discussion on some design considerations for the framework. This is followed by an outline of the scope of this design and a listing of framework requirements, with the various components necessary to fulfil these requirements described in turn. For clarity, this section is divided into design time and run time parts, to mirror the distinct operations of the framework. A description of the four models central to the framework's operation then follows. The models detailed are responsible for registering data sources, specifying the key domain entities, encoding SME and representing results. Furthermore, as client applications need to be able to communicate with the framework, details of the framework API are also documented.

Supporting non-technical domain experts in encoding their expertise is a central function of the framework; hence the design and requirements of the SME authoring tool to support this feature are detailed next in the chapter. Finally, the *expert-supported approach to data exploration*, which underpins the framework and models, is then described. This approach is divided into design time and run time sections, and details the end-to-end processes necessary to support casual users in using SME (encoded by non-technical domain experts) to explore multiple data sources from a domain.

3.2 Influence from State of the Art

In Chapter 2, a state of the art analysis took place on two specific research areas. Section 2.2.9 outlined the key findings from analysing *Complex Querying by Casual Users over Multiple Sources*, and Section 2.3.8 summarised the key findings from the review of *Domain Independent Tools to Support SME Encoding by Non-Technical Experts*. These findings have greatly influenced various aspects of the research described in this thesis. In particular they have impacted on the design of a framework to support casual users in exploring heterogeneous data sources from a domain, by leveraging SME encoded by non-technical experts. This section summarises how these influences impact the system design described in this chapter.

This thesis proposes a domain independent tool for non-technical experts to encode their experience and knowledge of a domain without the help of a knowledge engineer. Such an approach may not capture as much detail as a bespoke system supported by a knowledge engineer, however its simplicity and domain independent nature would make its adoption and widespread implementation more likely. As seen in Chapter 2, the most likely SME encoding approach to be adopted by non-technical domain experts is visual query building. Specifically a schema-based approach in a wizard/form interface appears to be the most intuitive way of supporting novice users. Furthermore, best practice in expertise encoding is for SME to be reusable in different installations, accordingly any queries created by domain experts in this tool ought to be encapsulated as rules in a reusable SME model.

As noted in Section 2.2.9, there is a lack of support for third party applications to incorporate complex querying mechanisms into their design, whilst maintaining the freedom to choose what interaction paradigm (facets, widgets, mashups, NLI etc) their users must follow. This often restricts applications to offering only a browsing or question answering approach to the data. Hence, a way of encouraging the framework to be used by a community of application developers is to have an API that offers their applications useful functionality. This method has proven popular in the development of mashups as well as applications that use Freebase[25] and SWSE (Harth et al. 2007). Some API's require the application developers to know specific query languages such as XQuery, SPARQL, or SQL, which may prove a barrier to wider use by application developers. Hence, this thesis proposes an API that only requires the SME selected by users to be passed to it, along with any associated parameters. This should make the API more accessible to developers, as they do not need to know separate query languages for different data sources.

As described in Section 2.3.8 of Chapter 2, there is a general lack of support for encoded SME to have its meaning tweaked or tailored by the end user, beyond inputting a keyword to focus a search. Though this kind of tailoring is not appropriate for some kinds of SME, in other cases it can be beneficial. For instance some faceted interfaces support this kind of explicit tailoring to a limited extent i.e. by changing the values associated with each facet (e.g. *price* greater than €50 AND less than €100) it provides a simple but effective way for users (or a client application referencing a user model) to make the term associated with each facet more appropriate at that time. Because SME can sometimes encompass quite subjective notions such as "cheap" or "near", the design proposed in this thesis will support users, where appropriate, to tailor values associated with semantic terms to their own interpretation.

As shown in Chapter 2, in terms of supporting the reconciliation of data from multiple heterogeneous sources, a useful approach in terms of scalability and integration costs is to allow each information source to co-exist in their own native format. Hence the design proposed in this thesis is for data sources to reside in their original location, with their schemas co-existing rather than being mapped to each other or to a canonical model. This pushes integration down to the instance level, as is done in Linked Data, and means that dataspaces of information sources can be constructed efficiently. Moreover, the design will support the addition of new sources to any dataspace, as well as the reusability of sources in different dataspaces.

[25] http://www.freebase.com

3.3 Design Considerations

How domain specific search can be very useful in supporting users in finding relevant information from a particular domain has already been discussed. However, unless the frameworks they are based upon are domain independent, and can be repurposed to different areas, their widespread adoption will be limited due to the time-consuming nature of developing bespoke systems. This thesis will directly address this need by describing the design of a framework that can be repurposed for different domains.

Modular frameworks that allow key components to be plugged into separate installations are useful in allowing different people to make separate contributions to a framework. Such an approach is followed in this thesis, where different reusable models can be created and plugged into separate frameworks. Furthermore, the framework design will support the extensibility of models, so that SME and data sources (even of new data types) can be added to an installation at any time. This facilitates the evolution of richer relationships between data sources over time, and enables new perspectives to be accommodated in the future. Ultimately, by supporting extensibility the framework is more dynamic and less likely to be made redundant quickly.

3.4 Framework Requirements

This thesis addresses how *subject matter expertise may be effectively encoded by non-technical experts and then leveraged by casual users to assist exploration and querying of multiple data sources from a domain*. From the analysis of deficiencies within the state of art, the following requirements were derived to support a framework that would help address this question, and that would be innovative and novel within its field.

- Provide client applications, which support user exploration, with consolidated access to multiple sources (in various data formats), without prescribing a specific user interaction paradigm for the GUI, or for their developers to know separate query languages for the various data sources.
- Enable the SME leveraged by the client applications to be encoded by non-technical experts, and to be tailored to an end-user's interpretation.
- Enable casual users that appropriate and tailor this SME within the client applications, to send complex queries to multiple data sources via the framework.

These requirements will be complemented with the following best practice findings derived from the state of the art.

- Allow data to reside in its original location, and provide instance level reconciliation between the different data sources.
- Be a domain independent and modular framework that supports the reusability and movement of sources and SME between different installations.
- Support extensibility by enabling the addition of extra SME and of new sources (even of a different data type) to enrich the relationships within the domain.

This thesis proposes a framework that is divided into design time and run time components, in order to support these requirements. This is a similar approach as used by Expert Systems (Taylor & Lubkeman 1989), where the knowledge base and expert rules are constructed at design time (after the framework itself has been fully implemented), with end users interacting with the system at run time. This means that models and rules generated by domain experts or knowledge engineers only need to be created once at design time, with multiple users able to interact with them at run time. If new SME and data sources are added at a later stage, this is seen as a further iteration of the design time process. In essence, end users are the only stakeholders that are directly involved at run time, whereas design time processes may involve one or more domain experts, knowledge engineers or application developers.

The overall framework this thesis proposes is a middleware system that sits between client applications and the distributed data sources. This puts fewer restrictions on the data sources as they can remain in their original location. It also allows greater flexibility in the design of client applications, as the only restriction is to conform to the system's API. Furthermore, it means that multiple applications can communicate with a single installation of the framework.

3.4.1 Scope of Framework

This section details the scope of the framework design presented in this thesis. For instance, the proposed framework is not an end-user facing tool, so a specification of design features for client applications (apart from conforming to the framework's API) are beyond the scope of this research. This is because the design emphasis of the framework is on giving developers freedom in how they implement their applications**.** Indeed, several applications that use this framework are outlined in Chapter 5 of this thesis.

This thesis describes a framework that operates on top of existing information sources. Hence, sources that have been carefully edited and curated by data modelling experts are particularly useful due to their higher degree of accuracy and reliability[26]. The generation of new data sources or the cleaning of existing sources, while relevant to this research, is beyond the scope of this thesis. However, these important processes are complementary to the approach described. Interfaces to many of these sources (in XML, RDF etc.) are well defined, hence the focus of this thesis is on manipulating and exposing this data in a way that is meaningful and accessible to casual computer users of the domain. This is achieved by enabling consolidated access to multiple data sources and using domain expertise to mediate between end users and the information they are interested in.

In terms of SME encoding within the framework, this thesis defines the chief role of the non-technical expert's as one of generating rules out of domain properties registered to the framework. Non-technical experts are not expected to have mapped these domain properties from the original data sources to the framework. However if mappings from a particular source have already been generated, then this thesis proposes that non-technical domain experts can indeed generate SME without the assistance of a knowledge engineer. More specific details on the roles of different stakeholders can be found in Sections 3.5.1, 3.9.1.1 and 3.9.1.2.

With regards to data accessed by the framework, it is assumed that is accessible within a reasonable latency. As seen in Chapter 2, high latencies can be problematic for systems, especially when accessing remote sources. This can greatly affect user satisfaction. Improving remote access speeds is beyond the scope of this thesis, however advances in this ongoing research area will benefit all distributed systems in general. Details of a performance evaluation of the framework with real world data sources are given in Section 5.7 of Chapter 5, with this experiment assessing whether the internal performance of the implemented framework is adequate.

[26] Examples of such publically available data sets include UK government statistics (http://data.gov.uk/about), USA government statistics (http://www.data.gov/) and Fingal County Council statistics (http://data.fingal.ie/about/).

3.5 Framework Description

The analysis of the state of the art led to a number of framework requirements as well as some design considerations and assumptions. This section describes how these led to the design of an overall framework, and gives an overview of its various components and models. It is divided into design time and run time sections to mirror the operation of the framework.

3.5.1 Design Time Framework

Figure 3-1 shows how the design requirements described in this chapter filtered down into issues that needed to be addressed in the design time framework. Essentially, it was necessary for the framework to support:

1. communication with different data sources from a domain.
2. storing of SME encoded by non-technical experts.
3. advertising of this SME to third party applications so that end users could leverage the SME in their data explorations.
4. the extensibility and reusability of SME and data source registration, as well as domain independence.

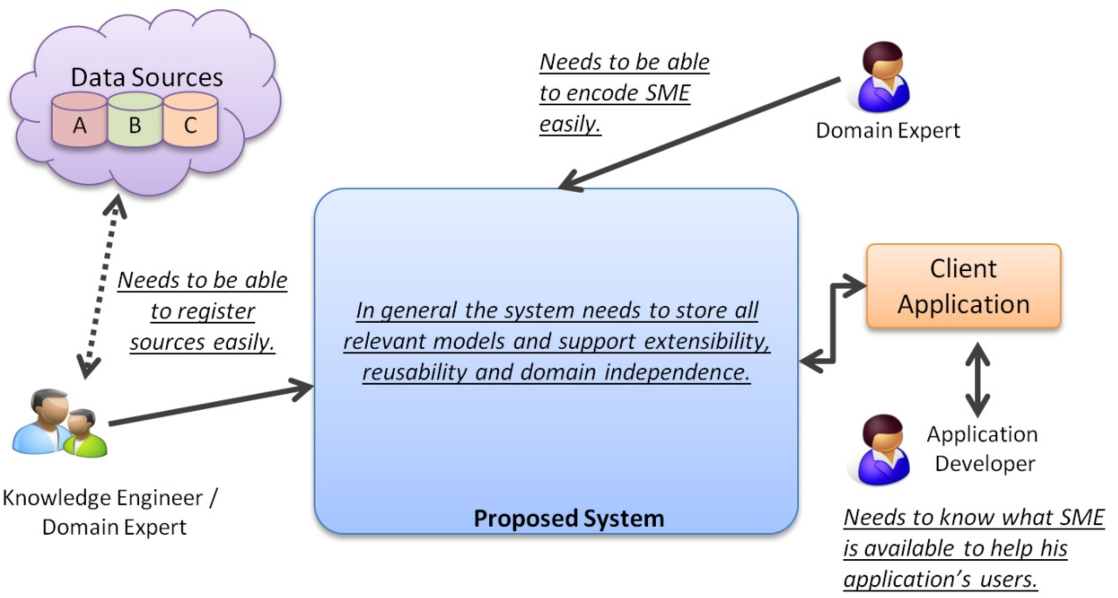

Data Sources
A
B
C
Needs to be able to encode SME easily.
Domain Expert
Needs to be able to register sources easily.
In general the system needs to store all relevant models and support extensibility, reusability and domain independence.
Client Application
Application Developer
Knowledge Engineer / Domain Expert
Proposed System
Needs to know what SME is available to help his application's users.

Figure 3-1. Issues Relating to the Design Time Framework

How these concerns were subsequently addressed in the design time framework is highlighted in Figure 3-2. The Reconciliation Engine is the name given to the central component which stores all relevant models and orchestrates the other components.

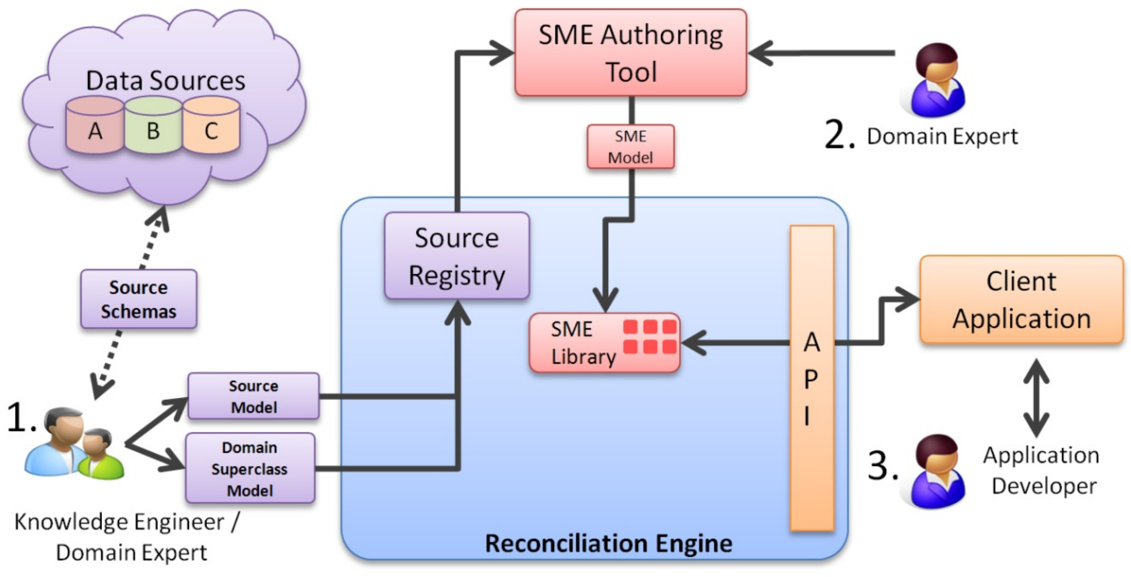


Figure 3-2. Design Time Framework

At position one in Figure 3-2 the knowledge engineer examines the schemas of data sources and any metadata from a schema of interest gets transcribed into a Source Model. In this thesis the registration of domain schemas is differentiated from the encoding of domain rules by experts who select properties from these registered schemas. Registering a source schema is a mechanical mapping process from the original schema into a Source Model, which can be consumed by the Reconciliation Engine. Hence, in many instances the need for domain expertise in this process is not necessary. These Source Models must be reusable in different installations and are stored in the Reconciliation Engine's Source Registry along with the Domain Superclass Model. The Domain Superclass Model simply contains the key entities from the domain in question chosen by the domain expert and encoded by the knowledge engineer. In essence, any queries made by client applications to the Reconciliation Engine are looking to return instances of one of these Domain Superclasses. Domain Superclasses are also a key mechanism in integrating different data sources, as they allow each source to declare which shared entities they store information about. The Source Registry itself is a key component of the Reconciliation Engine and stores all the Source Models and Domain Superclasses.

At position two in Figure 3-2 the domain expert must encode his SME and have it stored in the Reconciliation Engine. A key requirement of the framework is to support non-technical domain experts in encoding SME; hence an accompanying SME Authoring Tool is necessary to support this. As mentioned previously, this thesis considers the process of creating rules from data properties as SME encoding, which is a distinct process from the registration of metadata schemas to the Reconciliation Engine. As SME generated is based on the data sources connected to the Reconciliation Engine, the domain experts must know what sources and metadata are available to them. Hence, the Source Registry which contains each data source's key details is inputted to the SME Authoring Tool and presented to the domain expert. The expert then uses this tool to generate SME referred to in this thesis as semantic attributes. Each semantic attribute generated by the domain expert encapsulates rules in the form of queries that can traverse the related information source. For example, if the data source is in XML format, the semantic attribute will encapsulate an XQuery.

Another key requirement for the framework is to promote reusability and extensibility of SME, so the Semantic Attribute Model that encodes the semantic attributes must support both of these features. It is also important that the Semantic Attribute Model can support the expertise to be tailored by end users to their own requirements. Finally, any Semantic Attribute Models generated are stored in the Reconciliation Engine so that they are easily accessible by client applications. Hence, a Semantic Attribute Library component is part of the Reconciliation Engine. This stores the raw Semantic Attribute Models as well as generating the corresponding executable code for each. The executable code contains the queries encapsulated within the Semantic Attribute Model and ensures that client applications can access the SME efficiently at run time.

At position three in Figure 3-2 an application developer needs to know what semantic attributes are available so that he can create an appropriate user interface to support data exploration by the applications users. Hence an API is necessary in the Reconciliation Engine to allow requests for all the semantic attributes available. This API also supports multiple client applications in accessing the Reconciliation Engine simultaneously.

3.5.2 Run Time Framework

Figure 3-3 shows the how some of the key requirements detailed for the overall system filtered down into issues that needed to be addressed in the run time framework. Essentially, it was necessary for the system to support:

1. end users who send complex queries, based on Semantic Attributes, via client applications.
2. the forwarding of individual queries encapsulated within the Semantic Attribute Models to the relevant data sources in the appropriate query language.

3. the reconciliation of results from the separate data sources at an instance level.
4. the return of a consolidated set of results to the client application for rendering to the user.

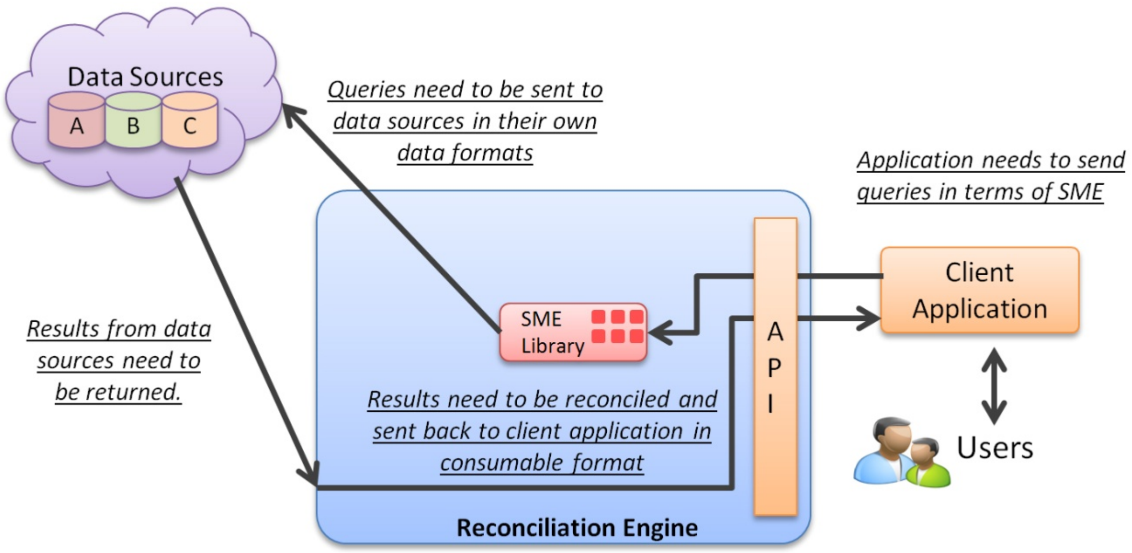

Data Sources
A
B
C
Queries need to be sent to data sources in their own data formats
Application needs to send queries in terms of SME
Client Application
SME Library
A P I
Results from data sources need to be returned.
Results need to be reconciled and sent back to client application in consumable format
Users
Reconciliation Engine

Figure 3-3. Issues Relating to the Run Time Framework

How these concerns were subsequently addressed in the run time framework is highlighted in Figure 3-4. The Semantic Attribute Library and the API are the only components within the Reconciliation Engine that are actively used at both design time and run time.

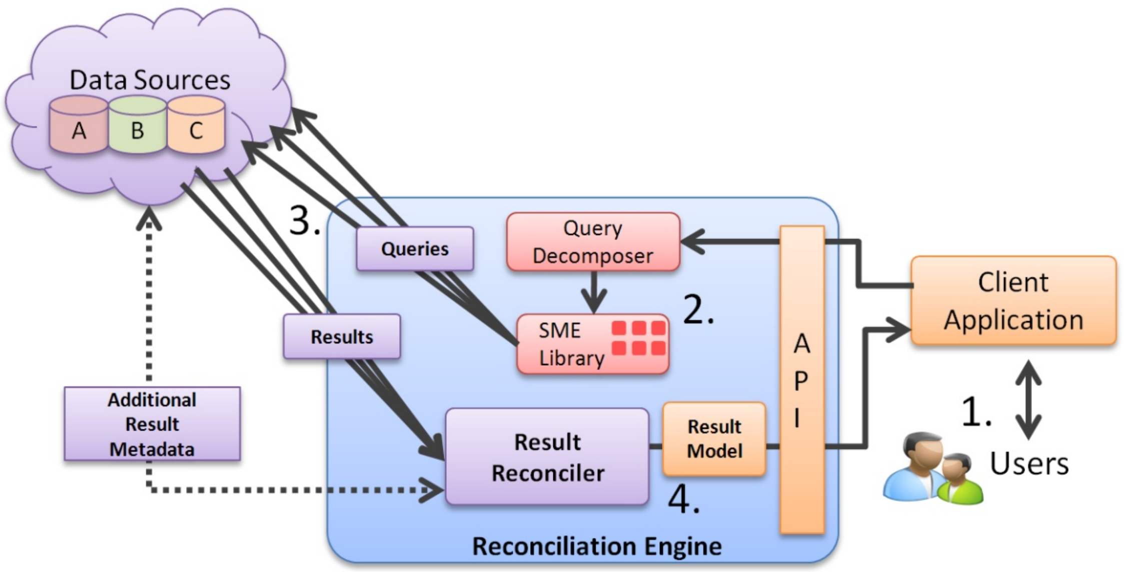


Figure 3-4. Run Time Framework

At step one the users engage with the client application using its GUI to generate a query in terms of the semantic attributes stored in the Reconciliation Engine. How users form these queries is entirely up to the application developers, as long as the queries sent conform to the Reconciliation Engine's API. Because the Reconciliation Engine employs a parameter based API, it makes it very easy for it to be used by application developers, as they do not need to know any data source query languages such as XQuery, SPARQL or SQL which some frameworks require. Instead they simply have to pass on the semantic attributes selected by users along with any associated operators and parameters.

At step two the inputted query is received by the Query Decomposer whose job is to break down the query into separate semantic attributes and then trigger the associated executable code in the Semantic Attribute Library. If the semantic attribute has not been tailored by the end user, then the query gets sent directly to its corresponding source. If semantic attributes have been tailored, then the parameter values specified by the user are plugged into the query before it is sent on to the relevant data source.

At step three each data source processes the query and sends back an individual result set containing the ids of instances of the relevant superclass. At step four the result ids from each individual semantic attribute set get consolidated together into a final result set which gets sent to the client application. This occurs in the Result Reconciler component. The default mode is to send only the result identifiers back to the client application, but it is also possible to send back additional metadata specifics about each instance, such as the values that triggered the semantic attribute rules.

The final result set is then converted into a Result Model which is passed back to the client application via the API. This Result Model can then be presented textually, or else visualised in some way to help the user interpret the information better. This is completely the prerogative of the client application and the framework has no bearing on this. After the results are displayed, the user can then tweak their queries iteratively and re-run them, or alternatively explore another area of the domain completely.

3.5.3 Summary of Framework Components and Models

Table 3-1 summarise the various components and models that are used in the design time and run time parts of the framework.

Table 3-1. Summary of Framework Components and Models

Name	Function	Design Time	Run Time
Reconciliation Engine	The overall engine that co-ordinates all the other components and models.	Yes	Yes
Source Model	Contains all the details for an individual source wishing to communicate with the reconciliation engine.	Yes	
Domain Superclass Model	Describes a key entity from the domain. Any query to the reconciliation engine must look to return instances of these superclasses. The superclass is also used as an integration mechanism between different data sources. Where it is possible to transform instances of specific superclasses into instances of another superclass is also described in this model.	Yes	
Source Registry	Stores all the Source Models and Domain Superclasses.	Yes	
SME Authoring Tool	Visualises the Source Registry and supports non-technical domain experts in generating SME.	Yes	
Semantic Attribute Model	Encoding of SME from the authoring tool	Yes	
Semantic Attribute Library	Stores all the Semantic Attribute Models created by experts.	Yes	Yes
Client Applications	Any applications that want to use the reconciliation engine to support their users in data exploration.	Yes	Yes
API	Facilitates design time and run time communication between client applications and the Reconciliation Engine.	Yes	Yes
Query Decomposer	Decomposes an incoming query into its constituent semantic attributes, inputs tailored values if necessary and sends individual queries to the requested sources.		Yes
Result Reconciler	Reconciles the individual result sets into a final result model.		Yes
Result Model	Represents the results in a form that can be parsed by client applications and presented to the end user.		Yes

3.6 Data Models

This section describes each of the four models used within the framework in turn. These models are the:

- Semantic Attribute Model

- Source Model
- Domain Superclass Model
- Result Model

3.6.1 Semantic Attribute Model

As highlighted in Section 3.5.1, the Semantic Attribute Model is a key component of the framework that describes the SME defined by experts which is leveraged by users in client applications. Because data is stored in many different formats and because it is useful to explore distributed sources in a consolidated fashion, it was necessary to construct a domain independent, extensible and flexible model for encoding a semantic attribute. Thus the main features for a Semantic Attribute Model are to:

- allow aggregation of metadata into parameterised rules that describes a domain concept or characteristic.
- be compatible with multiple common data formats and query languages.
- be extensible for new data formats.
- support expert defined rules as well as the tailoring of template rules.
- support consolidation of data at an instance level
- be self contained and able to combine with other semantic attributes to form more complex aggregated SME units
- be reusable in different installations to help mitigate against their manual nature of their construction

Semantic attributes are defined as *discrete units of domain expertise that can be combined together and tailored to support user exploration of an information domain*. They are created at design time and stored within the Reconciliation Engine (specifically in the Semantic Attribute Library) along with corresponding executable code. At run time, if a particular semantic attribute is accessed in the Reconciliation Engine, then its executable code, which contains the expert generated rule, queries its original data source. If this semantic attribute has been tailored by the end user in the client application, then the variables passed into the Reconciliation Engine are plugged into the query before being sent.

Semantic attributes typically act as abstractions and simplifications of the raw data, and are intended to make it more accessible for the ordinary, non-expert user. For instance, semantic attributes can encompass subjective characteristics such as *nearness, popularity* and *expensiveness,* as well their more objective values such as *distance in miles*, *number of records sold* and *price*. A semantic attribute may contain just a single metadata element or it may combine a number of metadata elements into a single semantic attribute, e.g. combining the elements *bitrate, sample rate* and *file type* into a single semantic attribute *audio file quality*. Only metadata from a single source can used to create a semantic attribute rule. This was a design decision that results in a simpler and more elegant model than one which encompasses metadata from multiple sources. The limitation of only addressing an individual source is mitigated by the fact that once a semantic attribute is generated, it can then be combined with semantic attributes relating to different sources (as described in the next section), thus enabling data from multiple sources to interact in a consolidated fashion.

Semantic attributes can be classified into one of three types; *expert*, *template* and *hybrid*. Table 3-2 describes the characteristics of each type of semantic attribute.

Table 3-2. Type of Semantic Attributes

Type	**Characteristic**	**Example**
Expert	A concept that the expert prevents the end user from tailoring.	Can vary from the quite objective e.g. *"Number 1 US Singles"* would only return US singles that reached number one in the US charts, to more subjective notions such as *"Popularity"* that the expert prefers not to be tailored in any way.
Template	A concept that must be tailored by end users and contains no expert defaults	The semantic attribute *"Music Genre"* would allow the user to search for specific instances such as *"Folk", "Rock"* or *"Jazz"* and the semantic attribute *"Contains Chemical Element"* would allow the user to search for substances that contain specific elements such as "*Hydrogen"* or "*Oxygen"*.
Hybrid	It contains expert default rules as well as values that can be tailored by the end user.	A sound engineer, in the context of his studio work, may use the expert defaults of *a "High quality audio file"* which insists on files containing uncompressed raw audio of 48,000Hz or higher. However, in the context of his own home listening, the same semantic attribute could be personalised by him to include any compressed MP3 files above a minimum value bitrate.

All semantic attributes can also be sub-categorised into a number of separate ranges or parameters e.g. the semantic attribute *Price* could be divided into {Expensive - Average – Cheap}, and *Weight* into {Under Weight - Normal Weight - Over Weight – Obese}. This categorisation allows non-experts to access information without detailed knowledge of the domain.

In terms of the Reconciliation Engine, semantic attributes are encoded within the Semantic Attribute Model. The exact items that the Semantic Attribute Model needs to define are listed in Table 3-3.

Table 3-3. Items required in the Semantic Attribute Model

Item	**Function**
Name	Name of the semantic attribute
Type	Type of the semantic attribute (*expert*, *template* or *hybrid*)
Location	Address of underlying data source
Superclass	The semantic attribute returns instances of this Domain Superclass
Variable Name(s)	If this is a *template* or *hybrid* Semantic Attribute, one or more variables are required for its template rule(s). These must have individual names.
Variable Type(s)	Each variable must have a corresponding Data Type
Parameter Name(s)	A semantic attribute can have multiple parameters. Each must have a different name and contain *expert* and/or *template* rules depending on the semantic attribute type.
Expert Rule	This is a parameter's *expert* rule which must be in a query language compatible with the underlying data source.
Template Rule	This is a parameter's *template* rule which must be in a query language compatible with the underlying data source and contain variables (defined above) that can be tailored by the end user.

New semantic attributes can be added to the framework at any time by different users. This facilitates teams of experts in performing collaborative work, and enables different expertise to be exposed across the same domain. This diversity of expert perspectives encoded as semantic attributes empowers end users to pick and choose the semantic attributes that are best suited to their needs. In some ways this is analogous to choosing a specific critic for guidance in a domain you have interest in, but in which you are not an expert. Thus subjective critiques on movies, sport, politics, finance etc. can be appropriated by the end-user to help their exploration, but more importantly they can be tailored to better match individual preferences.

3.6.1.1 Semantic Attribute Queries

Each semantic attribute is an atomic unit that can be joined together with standard logical operators by the user, in order to form more complex queries tailored specifically to their needs. These are called Semantic Attribute Queries, and they typically combine several semantic attributes together e.g. Return all *Songs* that are *very popular* AND that are *in a high quality audio format* AND that are either in the *blues genre* OR *jazz genre*. By tailoring the component semantic attributes it enables users to specify, if they wish to, what their interpretation is of a *high quality audio format* or a *very popular song* etc. In turn, this helps to bridge the semantic gap between end users and low-level data, as it supports users in exploring such data using semantics that are meaningful to them. Each semantic attribute can also include default values defined by the domain expert that allow informed queries to be run quickly without any tailoring. Figure 3-5 describes the information contained in the Semantic Attribute Query data stack at each of its four levels.

There are a number of advantages to incorporating the Semantic Attribute Model into a framework that enables expert-supported data exploration. Firstly this data model can support experts in defining subjective and objective SME that end users can employ to help their exploration of an information domain. The Semantic Attribute Model hides the underlying complexity of the raw metadata from the user and allows SME to be tailored to an end users own interpretation or current context. Importantly the Semantic Attribute Model does not specify the data format that information sources must be encoded in, which means that it does not limit the range of current (and future) data types that it is compatible with. This flexibility makes it applicable to a huge range of data sources. Finally, because each semantic attribute is modular, end users can combine individual semantic attributes into powerful compound queries. If provided with a sufficient range of semantic attributes, this feature gives users great freedom of expression while exploring a domain.

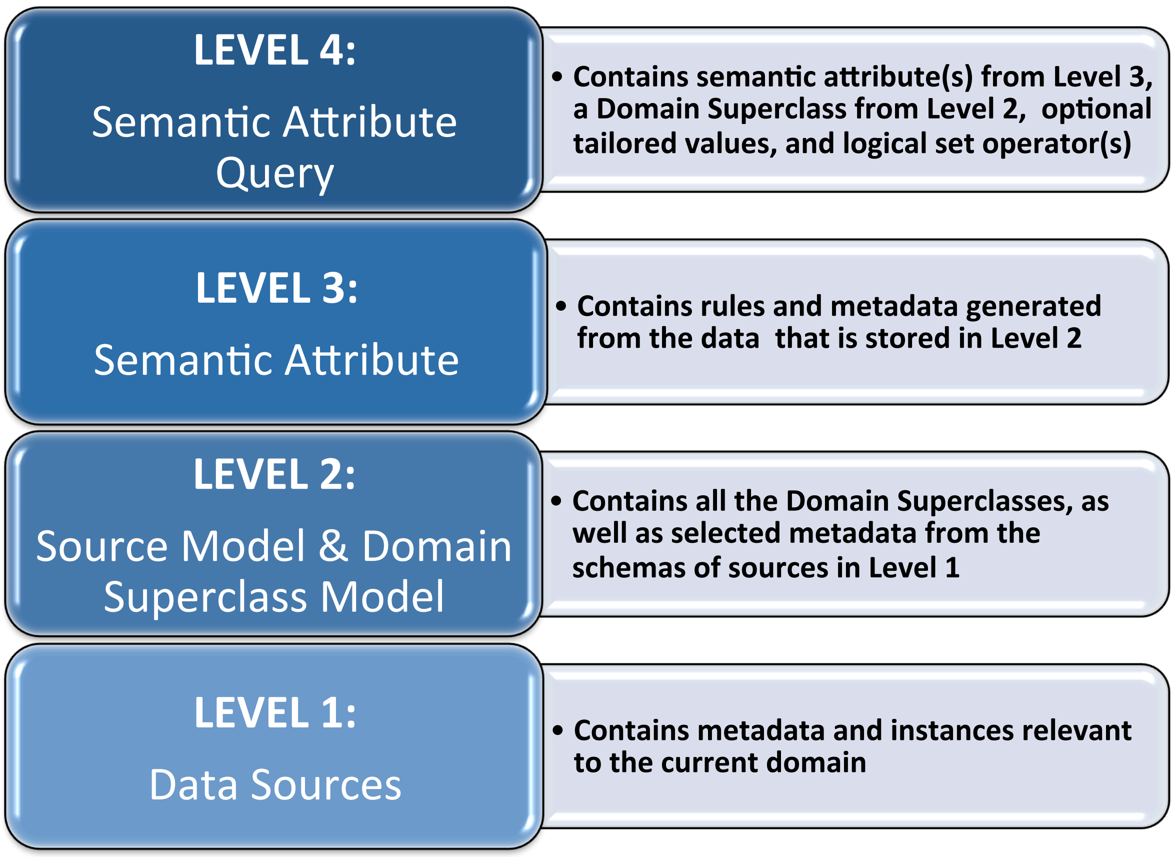

LEVEL 4:
Semantic Attribute Query
• Contains semantic attribute(s) from Level 3, a Domain Superclass from Level 2, optional tailored values, and logical set operator(s)
LEVEL 3:
Semantic Attribute
• Contains rules and metadata generated from the data that is stored in Level 2
LEVEL 2:
Source Model & Domain Superclass Model
• Contains all the Domain Superclasses, as well as selected metadata from the schemas of sources in Level 1
LEVEL 1:
Data Sources
• Contains metadata and instances relevant to the current domain

Figure 3-5. Semantic Attribute Query Data Stack

3.6.2 Source Model

Any sources that want to be accessible by client applications via this framework must reside in the dataspace set up for the domain. Dataspaces are collections of data sources that can be queried through a consolidated access point, despite full data integration not being initiated beforehand (Hedeler et al. 2010). It is more of a data co-existence approach and is achieved in this framework by registering a source model for each source with a Reconciliation Engines source registry. The Source Model describes key metadata from information sources, and its main function is to populate the SME authoring tool with elements about which domain experts can generate rules.

Each data format requires a different type of Source Model. Currently there are Source Models to accommodate data sources in three formats that are common for storing large volumes of data on the Internet (XML, RDF, and data accessible through a Web API). This section describes the Source Model for each of these three data formats.

3.6.2.1 XML Source Model

Table 3-4 shows the items that are necessary in the XML Source Model in order for an XML data source to be registered with the Source Registry within the Reconciliation Engine. The main function of this model is to populate the SME authoring tool with elements that the domain expert can use to create rules.

Table 3-4. Items Required in the XML Source Model

Item	Function
Source Name	Name of the data source
Location	Address of the XML database
Collection Name(s)	Name of database collection(s). Each collection has one or more elements. These are the building blocks used to generate rules by the domain expert.
Element Name	Name of the element. Each element must contain all the facets in grey.
Alias	In case the element's name is unclear away from its original context
Parent Name	Name of the elements parent node
Units	The units of the element if applicable (Km, Kg, etc.)
Superclass	The Domain Superclass the element is associated with
Identifier	The element that gives the unique identifier for this Domain Superclass instance.

3.6.2.2 RDF Source Model

Registering an RDF source (specifically a SPARQL endpoint) is a slightly more complex procedure than registering an XML source, but allows for more sophisticated queries to be generated than would be possible with XQuery. This is because SPARQL allows *joins* to be expressed implicitly simply by including two triple patterns that reference a common variable. This feature allows single semantic attributes based on SPARQL to reference multiple Domain Superclasses in a single *expert* rule. The main function of this model is to populate the SME authoring tool with predicates that the domain expert can then use to construct rules. Table 3-5 shows the items that are necessary in the RDF Source Model in order for the RDF data source to be registered with the Source Registry within the Reconciliation Engine.

Table 3-5. Items Required in the RDF Source Model

Item	Function
Source Name	Name of the data source
Location	Address of the RDF database or SPARQL endpoint
Prefix(es)	Namespace prefixes that the source predicates use
Superclass(es)	Any Domain Superclasses from the domain that this source contains, plus the SPARQL triple to return the identifier for each superclass instance.
Superclass Transform	SPARQL triple(s) to transform instances of one Domain Superclass to another.
Predicate Name	Similar to the elements in XML sources the predicates are the most important elements in RDF Source Models as these are what the domain expert generates his rules out of. Each predicate registered must describe all facets in grey.
Alias	Many predicates can be ambiguous so this makes the meaning of the predicate more apparent for the domain experts who must generate rules from them.
Units	The units of the predicate if applicable (Km, Kg, etc.)
Subject Superclass	Domain Superclass of the predicate's subject
Object Superclass	Domain Superclass of the predicate's object

3.6.2.3 API Source Model

Sources that can be accessed directly via a query language such as SPARQL or XQuery (whether hosted locally or remotely) do not require any more manual integration effort than is needed to construct a Source Model. This is important in supporting the rapid inclusion of information sources into a dataspace. Though information accessible through a native web service API requires a wrapper to make its data assessable to the Reconciliation Engine, this is not overly problematic. This is because any Web Services with well defined APIs can have wrappers developed relatively quickly. Moreover, creating a wrapper for a web service only needs be done once per source, and it can be reused in any other dataspace that wants to connect to that source.

The main function of the API Source Model is to populate the SME authoring tool with methods that the domain expert can construct into rules. Table 3-6 shows the items that are necessary in the API Source Model in order for data sources accessible through a native web service API to be registered with the Source Registry within the Reconciliation Engine.

Table 3-6. Items Required in the API Source Model

Item	**Function**
Source Name	Name of Web Service
Location	Wrapper Name within the Reconciliation Engine
Method Name(s)	Multiple methods can listed for each web service, and it is these which are used to generate rules by the domain expert. Each method must contain all the facets in grey.
Description	A description of the methods functionality
Superclass	The Domain Superclass instances the method returns
API Parameter(s)	Compulsory parameters and their associated data type needed to access the particular API call.
Wrapper Parameter(s)	Optional Wrapper parameters and their associated data types that are used to give more specific responses from the web service.

3.6.3 Domain Superclass Model

As described earlier, a Domain Superclass is a key entity from the domain in question which is chosen by the domain expert. In essence, any queries made by client applications to the Reconciliation Engine are looking to return instances of one of these superclasses. Domain Superclasses are also a key mechanism for integrating different data sources, as they allow each source to declare the shared entities about which they store information. The main functions of the Domain Superclass Model are to declare the key entities of interest in the domain, and to support the transforming of instances to other superclasses in order to facilitate more sophisticated querying. This process is described next.

When Semantic Attribute Queries (see Section 3.6.1.1) are sent to the Reconciliation Engine, the types of instances that are to be returned are specified by choosing one of the Domain Superclasses from the Domain Superclass Model. If users are restricted to making queries about a single superclass at a time, then no extra processing is needed. For instance the following query is related to the single superclass *Album* in the music domain:
*Return all **Albums** that are in the Jazz genre, that are shorter than 40 minutes and reached number 1 in the USA between 1950 and 1970.*

This type of query is perfectly sufficient for many applications. However, if a user is to make a query that spans multiple superclasses such as *Return **Albums** that contain UK number 1 **Songs** by **Music Artists** that have played in Ireland in the last year,* a mechanism for Domain Superclass transformation is necessary. Because this query returns *Albums*, all other superclasses in this query must have their results transformed into *Albums* also. Hence this query is broken into a number of steps:

1. Find all *Music Artists* that have played in Ireland in the last year.
2. Take these artists and find all the *Albums* that they have recorded.
3. Find all number 1 *Songs* in the UK.
4. Take these songs and find all the *Albums* to which they belong.
5. Intersect the *Albums* found in step 2 and step 4 to get the results.

Queries with multiple superclasses may result in many separate queries and result sets that expand and contract. However, it allows sophisticated queries to be formed easily by users, and accurate results to be sent back to the user. In order to perform such transforms automatically, superclass transformation data must be contained in the Domain Superclass Model. All items needed in the Domain Superclass Model are outlined in Table 3-7.

Table 3-7. Items required by the Domain Superclass Model

Item	**Function**
Name	The Name of the Domain Superclass
Input Superclass	Instances of this Domain Superclass can be transformed to the Domain Superclass in the Name field above. All facets in grey must be included for each Input Superclass.
Transformation Location	A location to perform the transform (Database, Endpoint, Service etc.)
Transformation Description	A description of the source that is performing the transform
Transform	A query or set of arguments to perform the transform
Output Superclass	Instances of this Domain Superclass will be outputted. Should correlate with the Name field.
Priority	If there are several locations to perform the same transformation, then the location with the higher priority gets preference.

In essences the transformation process involves the input of a set of instances of one superclass which are converted to a set of instances of the new superclass. Figure 3-6 show five superclasses with the arrows representing locations where superclasses can be transformed from one to the other. As you can see from the diagram *Music Artists* can be transformed to any of the other superclasses in just one step. However it is possible for superclasses to undergo multiple transformations such as *Songs* being converted into

Venues via *Music Artists*. Theoretically there is no limit to the amount of transformations that take place but the more transformations the longer the query processing will take. Hence the more direct transformation options between domain superclasses the better.

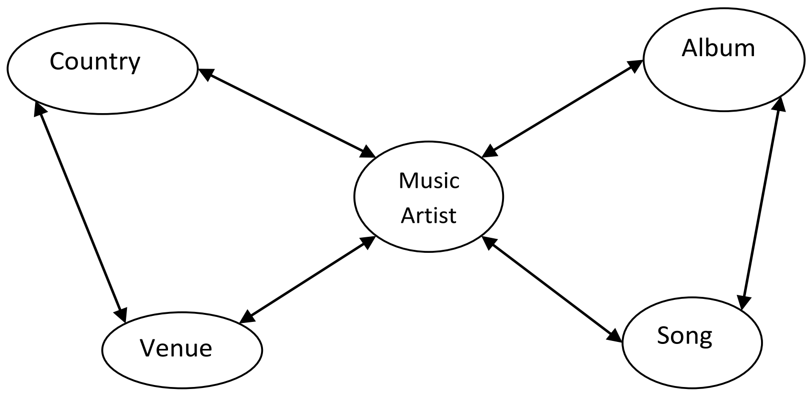


Figure 3-6. Graphical Representation of Five Superclasses and their Transformation Options

It is possible to have multiple transformation locations for the same transform so that the end user can specifically choose a transform location if they want more control over their query. As with other models in the framework, the Domain Superclass Model is completely reusable in any installation that has the same superclasses.

3.6.4 Result Model

The function of the Result Model is to contain all the results that are sent back to the client applications via the API. This model must be in a format that is easily parsed by applications for rendering to the end user. This model contains the name and number of instances returned by each constituent semantic attribute as well as the total number of instances returned by the full Semantic Attribute Queries. This feature informs the application if some individual semantic attributes gives results even if the Semantic Attribute Query itself does not, and thus enables the application to make more informed suggestions to the end-user regarding what to query next. This model also contains a list of identifiers for each instance in the result set, however it should be possible to augment these results with extra metadata on each instance to give the client application complimentary information to render. Table 3-8 shows the items required by the Result Model.

Table 3-8. Items required by the Result Model

Item	Function
Semantic Attribute Name	Name of each constituent Semantic Attribute
No. Results for each Semantic Attribute	Number of instances returned for each individual Semantic Attribute
Total No. Results	Number of instances returned for the Semantic Attribute Query
Identifier	An Identifier to distinguish each Semantic Attribute Query result
Instance Metadata	Optional metadata on each result instance to give client applications supplementary data to show users.

3.7 Reconciliation Engine's API

The Reconciliation Engine's API is the component that facilitates communication to client applications at design time and run time. At design time the API should allow client applications to query the reconciliation engine for a list of available semantic attributes according to their type (*expert*, *template* or *hybrid*) and data format (XML, RDF or data accessible through a native API). This API method is important because the application's users will interact with the Reconciliation Engine by selecting semantic attributes. The API method must make available the following information about each semantic attribute:

- The name of the semantic attribute.
- Type of semantic attribute
- The parameter names
- Any expert rules
- Any template rules
- Any variable names
- Any variable data types

By making this information available to the client applications a priori this API method enables them to tailor their user interface accordingly.

At run time the API should facilitate the sending of semantic attribute queries to the Reconciliation Engine and for a Result Model to be sent back. Because the client application is sending Semantic Attribute Queries it needs to send the following to the Reconciliation Engine's API:

- The Domain Superclass that the query is returning.

- The name of each semantic attribute.
- The parameter chosen for each semantic attribute.
- Whether each semantic attribute has been tailored or not.
- Any tailored values the user has inputted for a semantic attribute.
- The query language to be used for each semantic attribute.
- Whether each semantic attribute result set should undergo an intersection, union or difference operation with other semantic attribute result sets.
- Whether the results should be expanded to include supplementary metadata triggered by the semantic attribute.

The main advantage of this style of API is that the application developer does not need to know a specific query language in order to send complex queries to the Reconciliation Engine.

3.8 SME Authoring Tool

Because semantic attributes are hand-crafted by experts and not just automatically extracted from datasets, it is important to allow them to be created by non-technical experts in minutes. There are many ways of automatically extracting semantic features from data sets, however it can be argued that if domain experts can create semantic attributes quickly and easily then these can be of more value than automatically extracted features. Hence an SME authoring tool is necessary to directly support this, and is a key component of the design time framework. Such a tool would give the benefit of accurate human-created semantic attributes without the cost of a lot of manual effort.

The main design requirements for such an SME Authoring Tool were devised in reference to findings from state of the art (requirements 1-6) and the data models and components described in Section 3.6 (requirements 7-11). They are as follows:

1. Be accessible to users with no computer coding or information modelling background after minimal training.
2. Support a schema-based approach to query building using a wizard/form interface.
3. Automate as much of the creation process as possible to support these users.
4. Support rule generation for semantic attributes in multiple query languages.
5. Be extensible for new data formats.
6. Be able to query multiple data sources for the results to rules being generated, so as to provide instant feedback to domain experts as what the end user is to expect.

7. Allow all three semantic attribute types to be formed.
8. Work in tandem with the Reconciliation Engine.
9. Incorporate the Source Registry to display available metadata.
10. Group this metadata and make available for rule generation.
11. Generate the Semantic Attribute Model as output file.

Perhaps the key design decision in terms of the SME Authoring Tool's GUI, is that it should follow a schema-based approach to query building using a wizard system or forms. From the state of the art analysis, this approach appeared to show the most potential for supporting non-technical domain experts to encode SME as semantic attributes. Hence any SME Authoring Tool developed should ensure that this feature is incorporated in its user interface.

Figure 3-7 shows how the SME Authoring Tool interacts with the Reconciliation Engine. The Source Registry is inputted to the authoring tool and its metadata is used by the domain expert to generate rules encapsulated as semantic attributes. Each semantic attribute created in the authoring tool then generates a Semantic Attribute Model which is imported into the Reconciliation Engine and stored in the Semantic Attribute Library. While generating rules for each semantic attribute, it is possible for test queries to be sent to the data sources while the rules are being tweaked. The results sent back and displayed within the authoring tool help the domain expert to decide if a rule should be amended. This process is depicted as dashed lines in Figure 3-7.

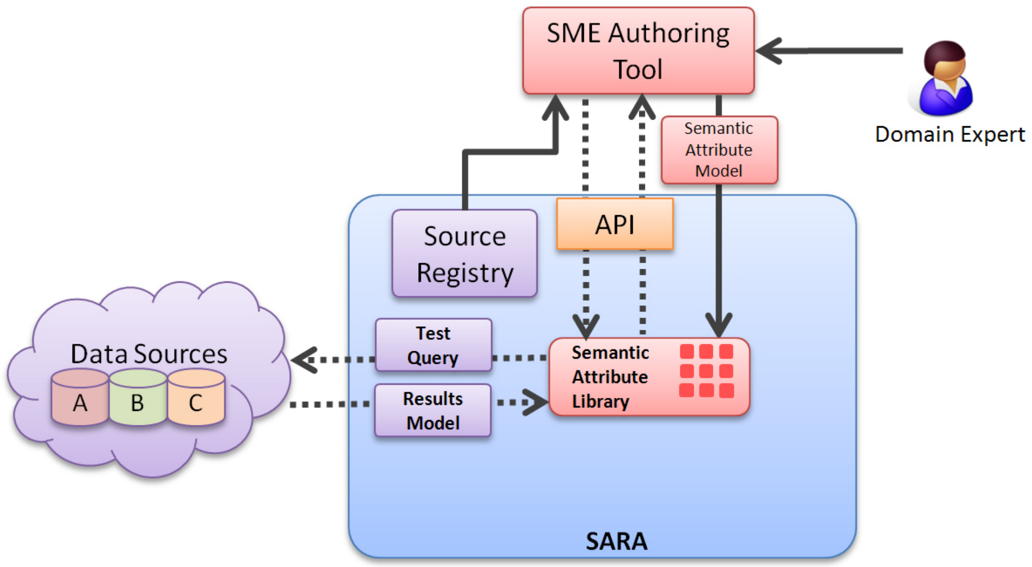


Figure 3-7. Interaction between the SME Authoring Tool and the Reconciliation Engine

From the expert's perspective it is important to stress that by using this authoring tool they should not need any knowledge of the underlying query languages (XQuery, SPARQL etc.) and should be able to construct rules in minutes. As described in Chapter 2, a balance needs to be struck between faithfulness to an underlying query language's expressivity and the ease of use of its GUI metaphor. Thus the level of expressivity offered will vary depending on the individual query language and how easy its constructs can be hidden from the casual user. However at a minimum, the SME Authoring Tool should enable domain experts to join multiple elements into a rule, and allow them to assign values using operators. The range of operators available will depend on the individual query language, but should at a minimum include *equals to*, *not equals to*, *greater than* and *less than*, which are intuitive to casual users. Finally, the end user should never be presented with any of this raw code, but merely uses the client application to select the semantic attribute he wants for his query. He can then choose to use the default *expert* rules or tailor the *template* rules via the GUI.

3.9 Expert-Supported Approach to Data Exploration

The *expert-supported approach to data exploration* defines an underlying process model that underpins the overall framework and models. This approach was developed in response to the generic KDDM (Knowledge Discovery and Data Mining) process model specified by Kurgan and Musilek. In their survey of the major process models in use by the KDDM community, Kurgan and Musilek specify a six step generic model which consolidates the information accumulated among the five major models (Kurgan & Musilek 2006).

The benefit of such a standardised process model becoming popularised within the KDDM community is that it would provide a common framework for researchers, thus providing cost and time savings. Because other disciplines like data exploration encounter many of the same issues as KDDM, it can also benefit from a structured process model. Thus this thesis proposes a seven step process model, called the *expert-supported approach to data exploration*, which is divided into separate design time and run time components to mirror the framework design previously outlined. These steps are listed in Table 3-9.

Table 3-9. Design Time and Run time Processes of Expert Supported Approach to Data Exploration

Design Time Processes	Run time Processes
1. Domain Understanding	4. SME Presentation & Query Generation
2. Source Selection	5. Query Routing
3. SME Encoding & Advertising	6. Result Consolidation
	7. Result Presentation

The following sections detail each of the steps in both the design time and run time processes and highlights how this approach benefits the various stakeholders listed in Table 3-10.

Table 3-10. Stakeholder Benefits of using Expert Supported Approach to Data Exploration

Stakeholder	Benefit
Non-technical and technical domain experts	With minimal training they can encode SME as rules operating over metadata.
Application developers	Can exploit the SME and consolidated view over an information domain to develop powerful applications.
End users	Are hidden from the underlying complexity and raw metadata of the sources, and can use the client applications to leverage SME while exploring information domains of interest.
Knowledge engineers	Can create a dataspace of structured and semi-structured sources by reusing existing models, or by registering new data sources.

3.9.1 Design Time Processes

The design time stages of this workflow process model consist of the three steps depicted in Figure 3-8. Each of these three steps follow each other linearly, however there can be feedback loops to any of the previous steps if there is a need for a process to be revisited. For instance, a feedback loop may be triggered from step 3 (SME Encoding & Advertising) to step 2 (Source Selection), if additional data sources are needed to encode more useful SME. The next section will describe each step in detail.

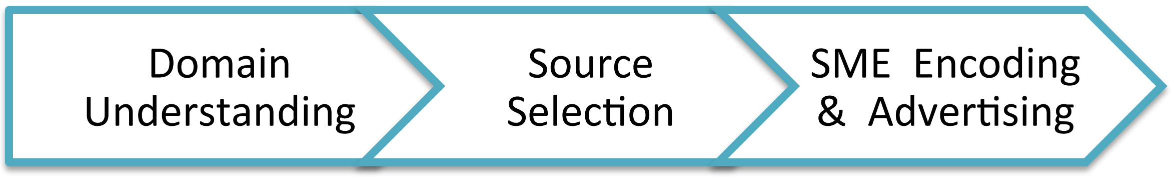


Figure 3-8. Design Time Processes of Expert Supported Approach to Data exploration

3.9.1.1 Domain Understanding

This step relates to understanding the goals of the end-user and aligning them in terms of Domain Superclasses. Though this initial process can be quite short, it greatly influences steps two (Source Selection) and three (SME Encoding & Advertising). The key person in this endeavour is the domain expert who must find out what is likely to interest end users and then select the Domain Superclasses accordingly. As described previously, these superclasses can be seen as any key entities from a domain about which a user would typically like to get information about. Choosing them is the first step in the *expert-supported approach to data exploration*.

There is no limit to the number of superclasses that can be selected for a domain and no need to define any relationships or properties for them, which can be an arduous task when creating domain ontologies. Furthermore, new superclasses can be added at any time without impacting on previous superclass choices, so domain experts are not limited to their initial selection. This is in keeping with the extensible nature of the framework.

Some examples of possible superclasses that could be selected in the Astronomy domain are *Planet, Star, Astronaut, Satellite* etc. Likewise in the music domain *Music Artist, Song, Album, Venue* etc. could be chosen by the domain expert. If data source A referred to *Artists*, data source B to *Groups* and data source C to *Singers*, the domain expert could decide that all these three concepts refer to the same Domain Superclass *Music Artist.* This simple association is vital in allowing multiple data sources to interact with each other, with any metadata relating these concepts in the original sources (name, age, nationality, albums etc.) needing no further integration.

Each superclass is independent of each other and the only task for the domain expert is to identify these key entities. They then get added to the Domain Superclass Model so that they can get assigned to data sources in the Source Selection step that follows. The final part of this process is optional, and consists of the knowledge engineer adding superclass transformation information to the Domain Superclass Model as outlined in Section 3.6.3. This gives client applications the ability to offer more sophisticated queries which contain multiple superclasses. Once a Domain Superclass Model is finished it is added to the Source Registry and each model can be reused in any other installation. This means popular Domain Superclasses and information on how to transform them to other superclasses do not have to be generated repeatedly.

3.9.1.2 Source Selection

The first part of this process is for the domain expert to identify data sources that contain useful information about the domain. In order for these sources to be accessible by client applications and consolidated with other sources, they must reside in the dataspace set up for the domain. This is achieved in the *expert-supported approach to data exploration* by registering each data source's Source Model with the Source Registry. Data sources suitable for selection are those that have structured or semi-structured data, and that contain instances of one or more Domain Superclasses. One of the key specifications in any Source Model, as identified in Section 3.6.2, is the Domain Superclasses that this particular data source returns instances of. These must be chosen from the selection of superclasses encoded in the Domain Superclass Model.

It is beneficial if sources contain a shared identification scheme for superclass instances, as this enables more sophisticated queries to be performed that reconcile data from different sources. Ideally these identifiers are dereferenceable URIs (Mei et al. 2008), which are increasingly being used, but such a comprehensive mechanism for identification is not a prerequisite for a source's inclusion. In addition to dereferenceable URIs, many useful identification schemes already exist in different domains and organisations. For instance below are some commonly used examples:

- Staff / Student / Social Security / Patient numbers
- Postcodes
- GPS coordinates
- Dates
- International Standard Book Numbers (ISBN)
- International Standard Serial Numbers (ISSN)
- EAN-13 Barcodes
- Life Science Identifiers
- Chemical Symbols

Though it is more useful if sources in a dataspace contain shared instance level identifiers for the Domain Superclasses, it is not a prerequisite for joining the dataspace. This is because it is still possible to find the union of result sets from individual data sources, despite them having different identification schemes.

Once a data source has its Source Model registered with the Source Registry it is assumed that the source is accessible within acceptable latencies and that the data is as accurate as is reasonably possible. Hence any data processing that is necessary to provide additional metadata about the source (perhaps calculating average figures etc.), or data cleaning to tidy up the data, takes place within this step. However, in many cases data sources can be taken "as is" and do not require any pre-processing in order to be useful.

Source Models can be added and removed to the Source Registry at any time, which is vital in making the creation of bespoke dataspaces as flexible as possible. Moreover each source's Source Model can be reused in any other installation, which means that the generation of a new dataspace out of a pre-configured Source Model is almost instantaneous. Because a lot of domains contain certain sources that are widely referenced, it is likely that many dataspaces will end up reusing these popular sources that have existing Source Models. Some specific data sources can then be added to complement them.

The creation of Source Models is typically done by a knowledge engineer or someone with a computer science background in consultation with the domain expert. However, if no data processing or data cleaning is required, it is conceivable that with the correct tools a non-technical domain expert could do this step themselves. Moreover, if the desired data sources have existing Source Models previously generated elsewhere, then the task becomes the trivial importation of these models.

3.9.1.3 SME Encoding and Advertising

Each source in the Source Registry will contain many references to instances of the Domain Superclasses (e.g. *The Beatles*, *The Rolling Stones* etc. as instances of *Music Artists*, and *Wembley Stadium*, *Madison Square Garden etc.* as instances of *Venues*). In each data source's schema they will have metadata relating to these instances and any metadata that is of interest (e.g. *song_duration* and *year_released* in a music database source) gets associated with one or more superclasses from the domain (e.g. *song_duration* with *Song* and *year_released* with *Album, Song* and *Artist*) in each Source Model. Through this process, different data sources with different schemas can co-exist in a dataspace without having to go through the time consuming and problematic process of being homogenised to a canonical model.

The metadata registered in each Source Model are the individual building blocks that the domain expert uses to encode rules as SME. The metadata in each Source Model is loaded into the SME Authoring Tool so that non-technical domain experts can generate these rules. For example, if the following elements *bitrate, sample rate* and *file type* were added to a Source Model, the domain expert could then generate a rule such as:

```
High Quality Audio File    =    bitrate > 319 KBs AND
                                sample rate > 44,099 KHz AND
                                file type = 'mp3'
```

Only metadata from a single source can be joined to form such a rule in a semantic attribute, however once a semantic attribute is generated it can then be combined with semantic attributes from any other source so this is not a major limitation. Because the domain expert can only create rules from the metadata contained in the Source Registry, it is vital that he has a wide selection to choose from in order to express rich SME. However because only minimal configuration of a Source Model is required to add new metadata elements from a source, and there is the option of reusing Source Models from other installations in the same domain, this should not become a major issue.

Any applications wishing to interact with the framework can do so through the formation of Semantic Attribute Queries. As mentioned previously, these queries consist of individual semantic attributes generated by the authoring tool and submitted to the Semantic Attribute Library. By accessing an API method, it is possible for client applications to know a priori what semantic attributes are present in the Semantic Attribute Library, and the functionality that they offer. This enables, the available semantic attributes be made available to end users in the GUI of the client application.

3.9.2 Run Time Processes

The run time processes of the workflow consist of the four steps depicted in Figure 3-9. These four steps follow each other linearly, however step 4 (result presentation) typically loops directly back to step 1 (query generation) when the user refines their exploration as a direct consequence of the results received for their previous query. This section describes each step in detail.

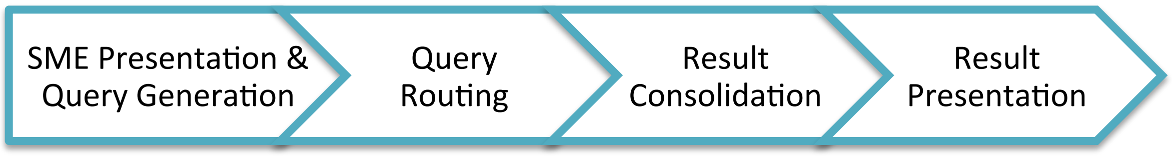


Figure 3-9. Run Time Processes of Expert Supported Approach to Data exploration

3.9.2.1 SME Presentation and Query Generation

An end user connects to the framework via a client application that uses the Reconciliation Engine's API. In this application the user chooses the semantic attributes in which they are interested (tailoring if desired) and joins them together with operators into a Semantic

Attribute Query, e.g. *Return all Artists from my iTunes collection that have Concerts Scheduled in the USA despite their most recent top 10 Album in the USA being more than ten years ago.* This is a combination of three semantic attributes (all artists in my iTunes, all artists with concerts scheduled in the USA and all artists with top 10 albums in USA before the year 2000) each potentially from a different source. Whether the client application presents the user with a simple query building mechanism or supports a more explorative approach using novel visualisations is irrelevant to the framework. It only requires that Semantic Attribute Queries are formed and passed to the Reconciliation Engine's API. If there is a huge array of semantic attributes available for a particular domain, the client application may use crowd sourcing (Brabham 2008) or user modelling techniques (Shen et al. 2005) to recommend specific semantic attributes for display to users. Such functionality is often applied in faceted browsing interfaces (Polowinski 2009), where the list of facets displayed to the user is dynamically adjusted depending on a number of factors.

3.9.2.2 Query Routing

When the framework receives the Semantic Attribute Query it decomposes it down into its individual semantic attributes and locates them in the Semantic Attribute Library. In the executable code associated with each Semantic Attribute Model the relevant query is located. If the semantic attribute has not been tailored by the end user then the query gets sent directly to its corresponding source (via a wrapper in the case of data sources behind a native API), otherwise the user's inputs get plugged into the query before it is sent onwards.

3.9.2.3 Result Consolidation

Once each source processes the query, it sends back an individual result set containing ids of instances of the relevant Domain Superclass. At this stage it is necessary to reconcile the individual result sets into one master set according to the operators sent in the Semantic Attribute Query. Using set operators, the result ids from each individual semantic attribute's result set get consolidated together into a final result set which gets sent to the client application. The default mode is to send only the result identifiers back to the client application as this is the most efficient technique. It is possible to send back additional metadata on each instance, such as the values that satisfied each semantic attribute's rule. However, this may result in slightly longer processing times due to the increased number of queries necessary.

3.9.2.4 Result Presentation

The Result Model is then created from the result set and sent back to the client application via the API. The Result Model can be parsed by the client application and results presented textually or else visualised to help the user interpret the information better. The degree of "exploration" that is offered to users directly depends on the user interface of the third party application. For instance, visualisation techniques may be used to provide novel ways to view result sets iteratively. However, how the results are presented to users is completely determined by the client application, and the framework has no bearing on this. After the results are displayed in the GUI, the user can then tweak their queries iteratively and re-run them, or else use the interface to explore a completely different area of the domain.

3.10 Summary

This chapter has described a novel approach to expert-supported data exploration and the design of a technical framework to support it. The requirements for this approach are based on the objectives outlined in Section 1.3 and the subsequent findings from the state of the art analysis. This chapter detailed all the necessary components and models to fulfil these objectives, as well as the seven processes that underpin the *expert-supported approach to data exploration*. The design detailed in this chapter will be used as the basis for a technical implementation of the Reconciliation Engine called SARA (Semantic Attribute Reconciliation Architecture) and its associated authoring tool SABer (Semantic Attribute Builder). The implementation of both these systems is described in the next chapter.

4 Implementation

The previous chapter described a novel approach to expert-supported data exploration and the design of a technical framework to support it. The requirements for this approach, and the necessary components and models to support the framework were also detailed. This chapter will discuss in detail how these design elements of the framework have been implemented in order to fulfil these requirements, as well as the technologies used. For the interested reader, Appendix A provides a brief overview of some of the common technologies associated with the encoding and retrieval of data and knowledge, e.g. XML, XQuery, RDF and SPARQL, which have been leveraged in the implementation described.

4.1 Introduction

The two major components of the framework outlined in Chapter 3, are the Reconciliation Engine and the SME Authoring Tool with which it works in tandem. These components have been implemented as the Semantic Attribute Reconciliation Architecture (SARA) and the Semantic Attribute Builder (SABer) respectively. This chapter first describes SARA and the models, interfaces and processes central to its operation. It then discusses the implementation of SABer and how it works in tandem with SARA. By following the design requirements outlined in the previous chapter, SARA and SABer make the *expert-supported approach to data exploration* a technical reality, and their implementations underpin client applications whose evaluations are described in Chapter 5. The chapter concludes with a multi-source case study, which highlights some of the technical features supported in the implementations of SARA and SABer.

4.2 SARA (Semantic Attribute Reconciliation Architecture)

The Reconciliation Engine is the central component of the entire framework, and based on the requirements outlined in Sections 3.5.1 and 3.5.2 of Chapter 3, it requires the following:

F1 Support communication with different data sources of various formats in a given domain.

F2 Store SME encoded by non-technical experts.

F3 Advertise this SME to third party applications, via an API that only requires parameters, so that end users can leverage the SME in their data explorations.

F4 Support extensibility and reusability of SME and data source registration, as well as domain independence.

F5 Enable end users to send complex, tailored queries based on semantic attributes, via client applications that send the associated parameters to an API.

F6 Allow individual queries encapsulated within the Semantic Attribute Models to be forwarded to the relevant data sources in the appropriate query language.

F7 Reconcile results from the separate data sources at an instance level.

F8 Return a consolidated set of results to the client application for rendering to the user.

The Semantic Attribute Reconciliation Architecture (SARA) was implemented in Java to incorporate all these features, and will be described under the following headings:

- Architecture and Technologies Employed
- Representation of Models
- Interface with Dataspace Sources
- Interface with Client Applications
- Parsing of Queries and Reconciliation of Results

4.2.1 Architecture and Technologies Employed

SARA is middleware implemented as a Java library that a client application must communicate with in order to interact with raw data sources. The data types that SARA currently supports are RDF and XML, in addition to data accessible through native Web APIs. These are commonplace data formats for storing large volumes of data on the Internet. Regardless, SARA is capable of integrating further data formats (relational databases, JSON etc.) at a later stage without impacting its current design (See Section 4.5) Figure 4-1 shows the design time architecture of the overall framework and closely follows the architectural design outlined in Section 3.5.1.

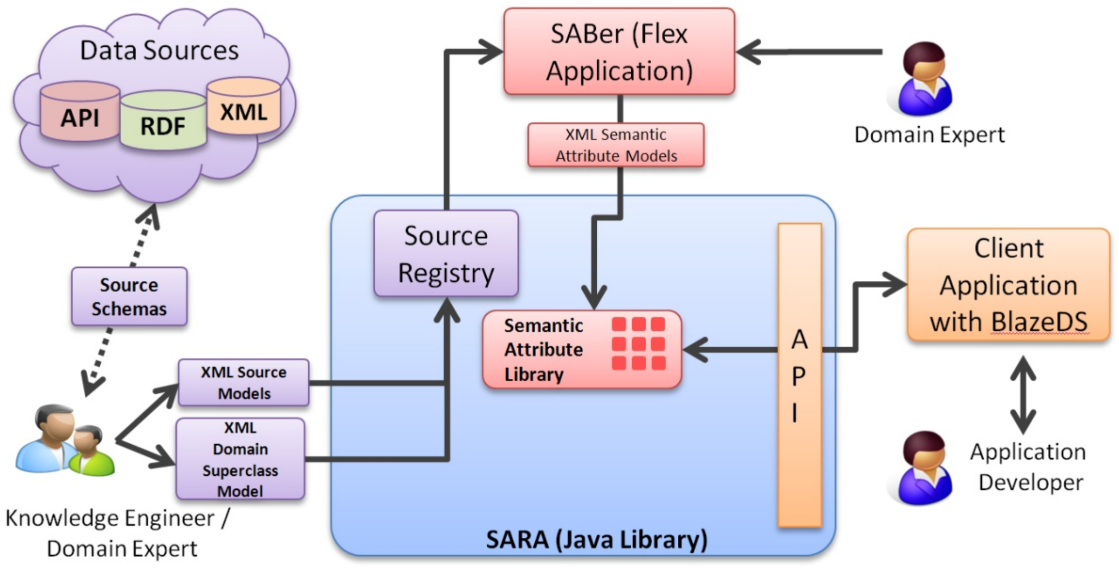


Figure 4-1. Design Time Architecture of Overall Framework

Figure 4-2 is a sequence diagram that displays a typical workflow in the design time architecture. As can be seen, all Data Sources have corresponding XML Source Models generated and sent for storage in the Source Registry. These models are stored alongside an XML Domain Superclass Model, which lists all the superclasses that the data sources contain instances of. The XML Source Models are also sent to SABer for rendering within its GUI. The next step is for a domain expert to generate SME using the SABer tool, which creates XML Semantic Attribute Models as an output. These models get stored in the Semantic Attribute Library alongside previously generated semantic attributes. If a client application wants to know what semantic attributes are available, it sends a request to the Semantic Attribute Library via SARA's API which returns an ArrayCollection of data on the Semantic Attributes. These semantic attributes can then be referenced by end users in the client application's GUI.

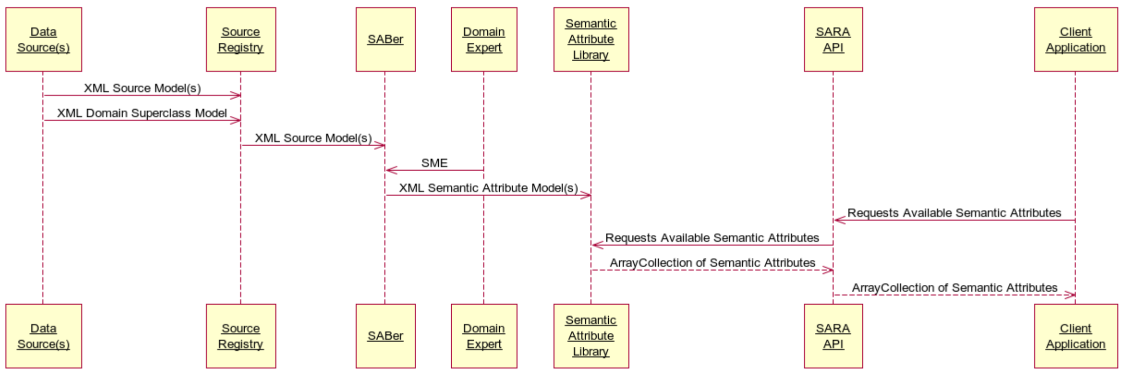

Data Source(s)
Source Registry
SABer
Domain Expert
Semantic Attribute Library
SARA API
Client Application
XML Source Model(s)
XML Domain Superclass Model
XML Source Model(s)
SME
XML Semantic Attribute Model(s)
Requests Available Semantic Attributes
Requests Available Semantic Attributes
ArrayCollection of Semantic Attributes
ArrayCollection of Semantic Attributes

Figure 4-2. Sequence Diagram for Design Time Architecture

Figure 4-3 shows the run time architecture of the framework which corresponds closely to the architectural design outlined in Section 3.5.2.

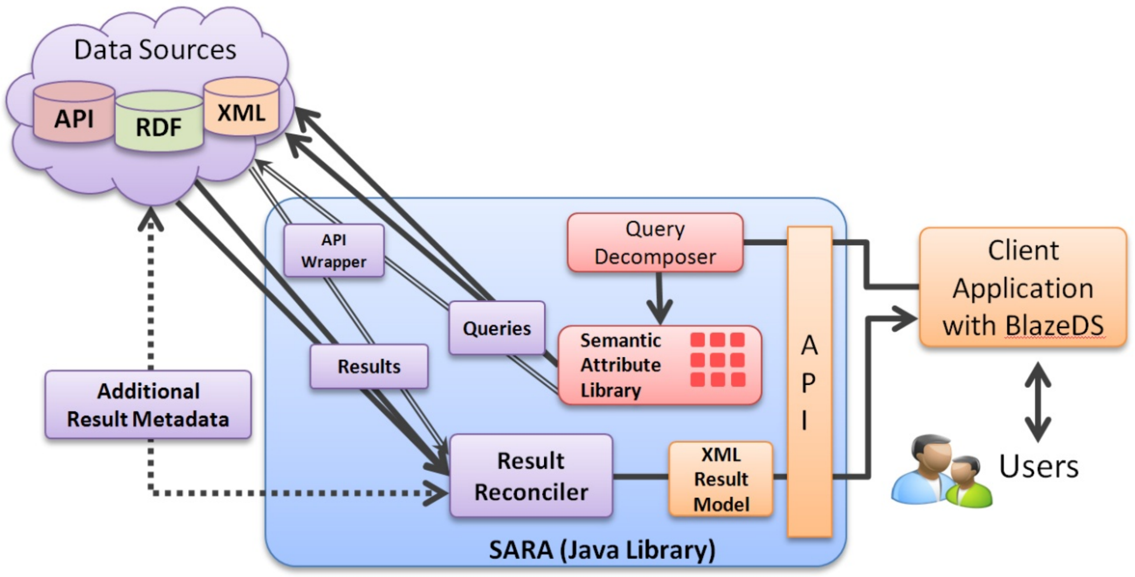


Figure 4-3. Run Time Architecture of Overall Framework

Figure 4-4 is a sequence diagram that displays a typical workflow in the run time architecture. It depicts a client application that is sending a Semantic Attribute Query to the Query Decomposer, via SARA's API. This particular Semantic Attribute Query references three separate semantic attributes, so the Query Decomposes locates each in the Semantic Attribute Library. Once a semantic attribute is found in the Semantic Attribute Library, the query encapsulated in it gets sent to its corresponding data source. In this scenario, it involves a SPARQL query going to a Linked Data Repository, an XQuery going to an XML database, and an API call going to a Web Service via a wrapper stored in SARA. Results from all these sources are sent to the Result Reconciler, which uses set manipulation to create a final unified results set. This gets transformed into an XML Result Model, which gets sent back to the Client Application for rendering via SARA's API.

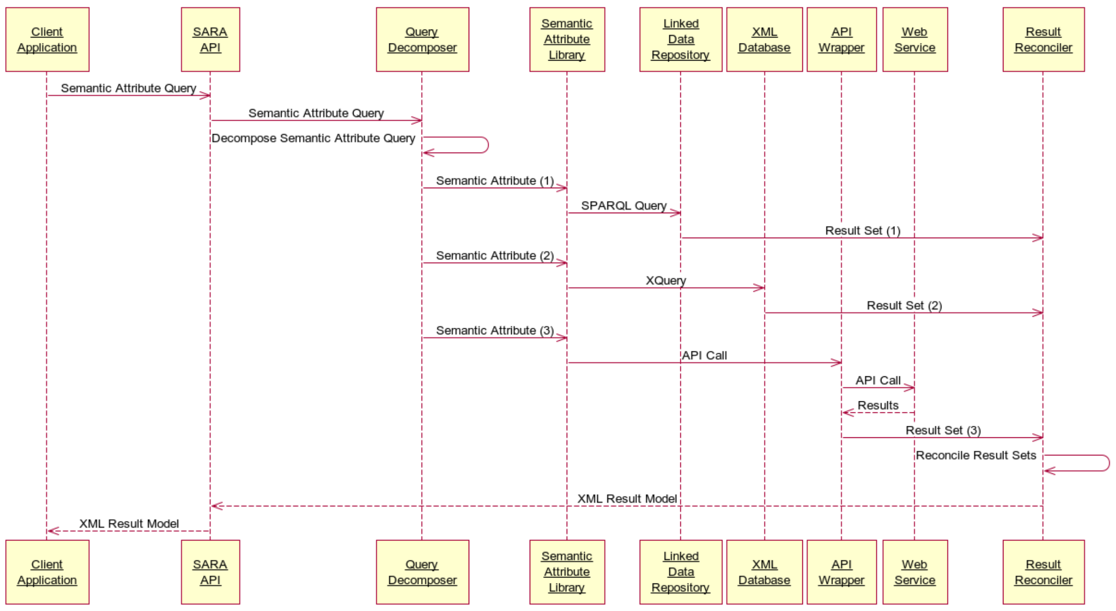


Figure 4-4. Sequence Diagram for Run Time Architecture

As is shown Figures 4-1 and 4-3, any client applications that use BlazeDS[27] can use SARA to access heterogeneous data sources. BlazeDS is a popular server-based Java remoting and web messaging technology that allows back-end distributed data to be sent in real-time to applications in many formats including Adobe Flex, Adobe Flash, AJAX and Adobe Integrated Runtime (AIR). BlazeDS is used by SARA to communicate with client applications; however SARA's API could also be offered using other technologies such as Web Services, making SARA's functionality available to an even wider variety of client applications. The architectures depicted in Figures 4-1 and 4-3 contain all the components and models outlined in the previous design chapter, with each element fulfilling a specific function. Each of the XML models shown in these diagrams, the way in which SABer interacts with SARA, and how the architecture supports the processes outlined in *the expert supported approach to data exploration* are described now detailed in this chapter.

[27] http://opensource.adobe.com/wiki/display/blazeds/BlazeDS

4.2.2 Representation of Models

As specified in the design chapter, SARA has four models that must be incorporated into its implementation. These models are the:

- Domain Superclass Model
- Source Model
- Semantic Attribute Model
- Result Model

Each of these models has been implemented in XML as it is a simple and widely used specification for encoding documents in machine readable form. This section will give an example showing how each model was implemented.

4.2.2.1 XML Domain Superclass Model

Figure 4-5 shows an example of a Domain Superclass Model. It follows the schema requirements laid out in Section 3.6.3 of the Design Chapter. If no superclass conversion is necessary from one superclass to another, then the only elements required in the model are the names of each superclass. Defining these superclasses is vital as they are the most fundamental building block of a Semantic Attribute Query (they specify what kind of instances the query is to return to the client application). Figure 4-5 describes how the superclass *Album* can be transformed into the superclass *Music_Artist* via a SPARQL endpoint (lines 4-28), and how the superclass *Song* can be transformed into the superclass *Music_Artist,* via a web service API (lines 29-39). This functionality is necessary for supporting some complex queries and is outlined in more detail in Section 4.2.6. Overall the Domain Superclass Model is central to supporting the SARA design features F1, F4 and F7 outlined in Section 4.2.

```
<Superclasses>
  <Superclass>
    <Name>Music_Artist</Name>
    <Conversion>
        <ConversionInput>Album</ConversionInput>
        <ConversionAddress>http://virtuoso.dbtune.org/sparql
        </ConversionAddress>
        <ConversionQuery>
        PREFIX mysp:<http://purl.org/ontology/myspace>
        PREFIX foaf:<http://xmlns.com/foaf/0.1/>
        PREFIX mo:<http://purl.org/ontology/mo/>
        SELECT DISTINCT ?id
        FROM <http://dbtune.org/myspace/>
        WHERE
          {
            ?result mysp:name ?id.
            ?album mysp:hasArtist ?result.
            ?album mysp:title ?1 .
            FILTER (?1 = " + Variable_1 + ").
          }
        </ConversionQuery>
        <ConversionFormat>RDF</ConversionFormat>
        <ConversionRationale>MySpace has a comprehensive list of
        official and unofficially released albums by artists in
        musicbrainz format
        </ConversionRationale>
        <Priority>1</Priority>
    </Conversion>
    <Conversion>
        <ConversionInput>Song</ConversionInput>
        <ConversionAddress>LastFM</ConversionAddress>
        <ConversionQuery>GetAllArtistsBySong</ConversionQuery>
        <ConversionRationale>Last.fm has a comprehensive list of
        official and unofficially released songs by artists in
        musicbrainz format
        </ConversionRationale>
        <ConversionFormat>API</ConversionFormat>
        <Priority>1</Priority>
    </Conversion>
  </Superclass>
</Superclasses>
```

Figure 4-5. Sample XML Domain Superclass Model

4.2.2.2 XML Source Models

This section describes the three XML Source Models that are used by SARA to register data sources with both native APIs and RDF or XML data sources. Overall, these models help support the SARA design features F1 and F4 outlined in Section 4.2, as well as responsible for populating the SABer authoring tool with data (see Section 4.3.1) so that users can generate semantic attributes.

4.2.2.2.1 XML Source Model for Sources with Native APIs

Figure 4-6 shows an example of a Source Model for a data source with a native API. It follows the schema requirements laid out in Section 3.6.2.3. The name of the web service that provides the API (line 1) and its corresponding wrapper name within SARA must be detailed (line 2). Multiple methods available through the API can be listed, which are used by the domain expert to generate rules. Each method contains its name, a description of its function, the type of superclass instances it returns and the API parameters that the web service requires (lines 5-16). In Figure 4-6, the method described (GetTopAlbumsByArtist) requires a single String argument which contains a music artist's name (line 11). Optional wrapper arguments may also be used to attain more specific responses from the web service API. Thus Figure 4-6 shows how it is possible to specify a range of albums to be returned from the service rather than having every available album sent back (lines 14-15). These arguments facilitate the domain expert to parameterise their rules into a number of ranges.

```
1   <Name>LastFM API</Name>
2   <Wrapper>LastFM</Wrapper>
3   <Methods>
4     <Method>
5        <Name>GetTopAlbumsByArtist</Name>
6        <Description>Returns albums according to their popularity on
7         Last.fm
8        </Description>
9        <Superclass>TrackTitle</Superclass>
10       <ApiParams>
11           <ArtistName type="String" units="N/A"/>
12       </ApiParams>
13       <WrapperParams>
14             <Highest_Popularity_Position type="int" units="N/A"/>
15             <Lowest_Popularity_Position type="int" units="N/A"/>
16       </WrapperParams>
17    </Method>
18  </Methods>
```

Figure 4-6. Sample XML Source Model for Sources with Native APIs

4.2.2.2.2 XML Source Model for RDF Sources

Figure 4-7 shows an example of a Source Model for an RDF data source. It follows the schema requirements laid out in Section 3.6.2.2 of the design chapter. It contains the name of the data source (line 1), the address of the RDF database or SPARQL endpoint (line 2), any namespace prefixes that the predicates use (lines 5-6), and any superclasses from the Domain Superclass Model that this source contains instances of (lines 8-17). The SPARQL code corresponding to each superclass in the model returns instances of this superclass from the data source. In its simplest form this code is just a single SPARQL triple in the form of *?result ?predicate_name ?id*, with *?predicate_name* being the only code changing from one superclass to another. For example in Figure 4-7 *?result foaf:name ?id* returns instances of the *Music_Artist* superclass (line 11) and *?result mysp:country ?id* returns instances of the *Country* superclass (line 15).

```
1   <Name>MySpace SPARQL Endpoint</Name>
2   <Location>http://virtuoso.dbtune.org/sparql</Location>
3   <Graph>&lt;http://dbtune.org/myspace/&gt;</Graph>
4   <Prefixes>
5         <Prefix>foaf:&lt;http://xmlns.com/foaf/0.1/&gt;</Prefix>
6         <Prefix>mysp:&lt;http://purl.org/ontology/myspace#&gt;</Prefix>
7   </Prefixes>
8   <Superclasses>
9         <Superclass>
10              <Name>Music_Artist</Name>
11              <Code>?result foaf:name ?id.</Code>
12        </Superclass>
13        <Superclass>
14              <Name>Country</Name>
15              <Code>?result mysp:country ?id.</Code>
16        </Superclass>
17  </Superclasses>
18  <Predicates>
19        <Predicate>
20              <Name>mysp:totalFriends</Name>
21              <Alias>Total friends on MySpace is</Alias>
22              <Subject>Music_Artist<Subject>
23              <Object>Value</Object>
24              <Units>N/A</Units>
25        </Predicate>
26        <Predicate>
27              <Name>mysp:country</Name>
28              <Alias>Country that MySpace artist is from</Alias>
29              <Subject>Music_Artist<Subject>
30              <Object>Country</Object>
31              <Units>N/A</Units>
32        </Predicate>
33  </Predicates>
34  <Transforms>
35    <Transform>
36          <SuperclassSubject>Music_Artist</SuperclassSubject>
37          <SuperclassObject>Country</SuperclassObject>
38          <SuperclassJoin>
39          ?Music_Artist mysp:country ?id.
40          </SuperclassJoin>
41    </Transform>
42  </Transforms>
```

Figure 4-7. Sample XML Source Model for RDF Sources

As can be seen in Figure 4-7, the predicate *mysp:country* has the alias *Country that MySpace artist is from* so that it is clearer to domain experts what this predicate actually represents (line 28). This predicate has the subject *Music_Artist* (line 29) as this is the domain superclass that has *mysp:country* as a property in this particular data source. Likewise, the predicate *mysp:country* has a corresponding object of a *Country* superclass (line 30), as these are the type of instances that this predicate returns from this data source. In the case of the *mysp:totalfriends* predicate, its subject is also *Music_Artist* (line 22), with its object being a specific value (the number of friends an artist has on the MySpace website) rather than another domain superclass. Thus "Value" is inputted instead of a superclass name (line 23).

The final part of the model shown in Figure 4-7 describes the transform information necessary to convert instances of one superclass to another (lines 34-42). In this instance it depicts the SPARQL triple necessary to transform *Music_Artist* instances into *Countries* (line 39). Any variable in a SPARQL triple that is referencing a superclass must have the same spelling, thus the superclass *Music_Artist* is referenced by the SPARQL variable *?Music_Artist*.

4.2.2.2.3 XML Source Model for XML Sources

Figure 4-8 shows an example of a Source Model for an XML data source. It follows the schema requirements laid out in Section 3.6.2.1 of the design chapter. It contains the name of the source (line 1), its location (line 2), any collections contained in the associated XML database (lines 3-17), and any elements of interest to the domain expert (lines 7-16). The data source in Figure 4-8 has only one collection. The elements registered are the most important as it is these that can be used to generate rules by the domain expert. Lines 8-15 of Figure 4-8 shows one of its elements (TrackDuration) but any amount can be listed in the model. Each element describes its original name in the data source, an alias in case its name is unclear away from its original context, its parent node, its unit type if applicable, the type of superclass it returns, and the element that gives the unique identifier for instances of the domain superclass.

```
1   <Name>iTunes eXist DB</Name>
2   <Location>xmldb:exist://localhost/exist/xmlrpc/db/SARA_0.1</Location>
3   <Collection>
4         <CollectionName>
5          John's iTunes Collection 01/01/10
6          </CollectionName>
7         <Elements>
8           <Element>
9               <Name>TrackDuration</Name>
10              <Alias>Duration_of_Song</Alias>
11              <ParentNode>//AudioTrack</ParentNode>
12              <Units>Seconds</Units>
13              <Superclass>Song</Superclass>
14              <ID>Song_Title</ID>
15          </Element>
16         </Elements>
17  </Collection>
```

Figure 4-8. XML Description of XML Source in Domain Registry

4.2.2.3 Semantic Attribute XML Model

Each semantic attribute created in SARA's authoring tool SABer should generate a Semantic Attribute Model that can then be imported into SARA. Figure 4-9 shows an example of a Semantic Attribute Model for a simple semantic attribute named "Popular Irish artists on MySpace". It follows the schema requirements laid out in Section 3.6.1 of Chapter 3 (e.g. the return superclass must be specified), and describes a *hybrid* semantic attribute that queries a SPARQL endpoint to find any Irish music artists with a minimum amount of MySpace fans. Because this is a *hybrid* semantic attribute, it provides both an *expert* rule and a *template* rule. The default *expert* rule specifies that the artist must be from Ireland, and have more than 50,000 MySpace fans (lines 14-25). When this rule is selected, the SPARQL query that represents it is sent to the data source encoded in the <SPARQLendpoint> element (line 4).

If the end user feels that this value in the *expert* rule is too high or too low, they can use the *template* rule (lines 28-39) instead to input what they feel is a more appropriate number of fans to be deemed "popular". When this rule is selected, the values that the user inputs for each of the variables (line 6-9) are plugged into the *template* rule (line 38), and then the completed query is sent to the data source. Hence, these *template* rules are vital for enabling end users to tailor semantic attributes to their own interpretation. This particular semantic attribute only has a single parameter (lines 11-41); however it is possible to have multiple parameters in each semantic attribute. By employing multiple parameters, semantic attributes are allowed to have different rules that create a range of different values (High, Medium, and Low etc.).

The Semantic Attribute Model is a critical part of SARA, and is the key link between the domain experts and the end users who are exploring a specific domain. Furthermore, it is likely that the number of Semantic Attribute Models in SARA will be many times larger than any other of the models. Hence it is vital that this model can be generated easily and does not need to be handcrafted in XML. This is one of the major motivations behind the SABer authoring tool described in Section 4.3. Overall, the Semantic Attribute Model is central to supporting the SARA design features F4, F5 and F6 outlined in Section 4.2. It is also the outputted model generated by non-technical domain experts in SABer.

```
<SemanticAttribute>
<Name>Popular Irish music artists on MySpace</Name>
<TypeOfSemAtt>Hybrid</TypeOfSemAtt>
<SPARQLendpoint>http://virtuoso.dbtune.org/sparql</SPARQLendpoint>
<ReturnSuperclass>Artist</ReturnSuperclass>
   <TemplateVariables>
      <VarName>Variable_0</VarName>
      <VarType>double</VarType>
   </TemplateVariables>
<Parameters>
 <Parameter>
  <Name>Default</Name>
  <ExpertRule>
     PREFIX mysp:<http://purl.org/ontology/myspace>
     PREFIX foaf:<http://xmlns.com/foaf/0.1/>
     PREFIX mo:<http://purl.org/ontology/mo/>
     SELECT DISTINCT ?id
     FROM <http://dbtune.org/myspace/>
      WHERE
       {
         ?result foaf:name ?id.
         ?result mysp:country 'Ireland'.
         ?result mysp:totalFriends ?1.
         FILTER (?1 > 50000 ).
       }
   </ExpertRule>
   <TemplateRule>
     PREFIX mysp:<http://purl.org/ontology/myspace>
     PREFIX foaf:<http://xmlns.com/foaf/0.1/>
     PREFIX mo:<http://purl.org/ontology/mo/>
     SELECT DISTINCT ?id
     FROM <http://dbtune.org/myspace/>
      WHERE
       {
         ?result foaf:name ?id.
         ?result mysp:country 'Ireland'.
         ?result mysp:totalFriends ?1 .
         FILTER (?1 > " + Variable_1 + ").
       }
   </TemplateRule>
 </Parameter>
</Parameters>
</SemanticAttribute>
```

Figure 4-9. Sample Semantic Attribute Model

4.2.2.4 Result Model

Figure 4-10 shows an example of a Result Model sent by SARA to a client application where only the identifiers of the superclasses are required. This is the most typical situation, and it follows the schema requirements laid out in Section 3.6.4 of Chapter 3. Figure 4-10 shows that the results come from a Semantic Attribute Query containing two semantic attributes (FilmDirectedBy and HighlyProfitableFilm); one with 37 results (line 4) and the other with only 5 (line 8). However, there are only three instances of the *Film* superclass that satisfy both of these semantic attributes (lines 11-13), thus theirs are the only identifiers listed. By showing how many results each individual semantic attribute returns, as well as the actual results, the client application is given potentially useful information to display while the user is formulating their next query. Overall this model is central to supporting the SARA design feature F8 as outlined in Section 4.2

```
<Results>
   <SemAtt>
      <Name>FilmDirectedBy</Name>
      <NumResults>37</NumResults>
   </SemAtt>
   <SemAtt>
      <Name>HighlyProfitableFilm</Name>
      <NumResults>5</NumResults>
   </SemAtt>
   <TotalResults>3</NumResults>
   <Result>The Godfather</Result>
   <Result>Jurassic Park</Result>
   <Result>Memento</Result>
<Results>
```

Figure 4-10. Sample XML Result Model

4.2.3 Interface with the Dataspace Sources

SARA itself does not store any of the data from the various information sources; instead the data resides at the source location until it needs to be queried. Any queries sent to these sources are generated by domain experts in SABer as part of the semantic attribute creation process and are encapsulated within the Semantic Attribute Model (see Section 4.2.2.3). Once a Semantic Attribute Query is sent to SARA it is decomposed into separate semantic attributes, then the related queries are extracted, and finally these queries are sent to the relevant data sources. In order to access RDF sources, SPARQL queries are generated and sent via Jena's[28] ARQ[29] query engine. XML data is accessed by generating XQueries which are sent to the sources via the XML:DB[30] API.

As mentioned in the Section 3.6.2.3, data that resides behind a native API can be accessed via SARA through a reusable bespoke wrapper. There is no restriction on how the wrapper interfaces with the API, and additional processing on the returned results can occur if desired. For instance, a web service might return the top songs of an artist if sent the name of the artist as a parameter, but always insists on returning 50 songs in a list from 1-50. A wrapper could do additional processing of the result list such as allowing users to specify

[28] http://jena.sourceforge.net/
[29] http://jena.sourceforge.net/ARQ/
[30] http://xmldb-org.sourceforge.net/

which range of songs to return (1-10, 35-40 etc.). The only stipulations for wrappers are that they are written in Java and adhere to the following guidelines:

- The Java class names for all wrappers in a SARA installation are unique.
- It is these wrapper names that are referenced by Source Models.
- Each wrapper must have a public method whose name concatenates "Call" with the wrapper's unique name e.g. the wrapper named "LastFM" must have a method called "CallLastFM"
- This method must return a HashSet of superclass instance identifiers
- This method's parameters must be:
    - *apiArg*s (Arraylist of the parameters needed by the web service's API method)
    - *wrapperArgs* (ArrayList of parameters that the wrapper uses for additional processing)
    - *APIMethod* (String identifier for the API method in the web service)
- The *APIMethod* parameter is used to identify which of the web service's API methods is to be called, with any arguments from *apiArgs* sent to it. The *APImethod* parameter should always begin with "Get".

All the interfaces mentioned in this section are central to supporting the SARA design feature F6 as outlined in Section 4.2

4.2.4 Storing Models

SARA must store three types of models, Domain Superclass Models, Source Models and Semantic Attribute Models. Domain Superclass Models are stored in the Source Registry in case a conversion between superclasses is necessary at run time (see Section 4.2.6 for more details). SARA also stores all Source Models in the Source Registry so that SABer can parse this XML, and display all the metadata options to the domain experts. Thus the Source Registry is central to supporting the SARA design feature F1 as outlined in Section 4.2. When a user finishes creating a semantic attribute in SABer, the XML Semantic Attribute Model is generated and is sent back to SARA to be stored in the Semantic Attribute Library. In order for this semantic attribute to be accessible by client applications the following must happen:

- SARA uses JDOM to parse the Semantic Attribute Model (JDOM is an open source Java-based document object model for XML) and extracts all the relevant data such as its name, variables, rules, superclass etc.
- The data extracted from the semantic attribute using JDOM is then concatenated with Java code to automatically construct a corresponding semantic attribute method. This method then gets appended to the end of its related master Java file.
- SARA currently has three such master files, one each for XML, RDF and API based semantic attributes.
- Each master file then has its class dynamically reloaded using a Java Proxy class[31], so that the new semantic attribute method is available and can be called by SARA if a client application wishes to access it.

The Semantic Attribute Library is thus the key component that supports SARA design feature F2 as outlined in Section 4.2.

4.2.5 Interface with Client Applications

As mentioned earlier, SARA currently supports communication between it and any client applications that use the BlazeDS remoting technology. An API has been developed for SARA that provides three methods for client applications. The first of these API methods is *QuerySARA* which is used at run time by a client application to send its Semantic Attribute Queries to SARA. This API method is central to supporting the SARA design feature F5 as outlined in Section 4.2. A description of the *QuerySARA* method follows:

QuerySara

```
public String QuerySara(String ReturnSuperclass,
ArrayCollection ChosenSemAtts)
```

Parameters:

ReturnSuperclass - a string detailing the instances of superclass that the user would like this Semantic Attribute Query to return. The choice is limited to those superclasses defined in the Domain Superclass Model.

[31] http://download.oracle.com/javase/1.4.2/docs/api/java/lang/reflect/Proxy.html

ChosenSemAtts - is an ArrayCollection of semantic attributes that the user has chosen in their semantic attribute query. Each semantic attribute itself is represented as an ArrayCollection and must be in the format described in Table 4-1.

Table 4-1. Parameters for a Semantic Attribute's Representation in the QuerySARA API Method

Pos.	**Name**	**DataType**	**Description**
0	sem_att_name	String	The unique name of the semantic attribute.
1	param_name	String	The parameter name chosen for the semantic attribute. If it is a *template* semantic attribute the parameter is named "default"
2	isTailored	String	Specifies whether the semantic attribute has been tailored or not. It is "false" if the user has just used the expert defaults and "true" if the user has tailored a rule.
3	tailored_args	Array Collection	Contains the tailored values the user has inputted. If a semantic attribute has not been tailored an empty array collection is sent.
4	query_type	String	Specifies the underlying query language to be used. Currently can be "XQuery", "SPARQL" or "API"
5	operator_group	String	Specifies which operator group the semantic attribute is in and hence what set operation should be performed on the result set. The options are: 1 *"MustHaveAll"* - results must have all these semantic attribute properties *2 "MustHaveAtLeastOne"* - Results must have at least one of these semantic attribute properties

			3 *"MustNotHaveAny"* - Results must not have any of these semantic attribute properties 4 *"MustNotHaveAll"* - Results must not have all of these semantic attribute properties
6	results_expanded	String	Specifies whether the Result Model is expanded to also contain the metadata triggering the semantic attribute's rule. If set "true" it will take longer to process results than if set "false" due to additional queries that need to be sent.

Returns:
An XML Result Model

In relation to the *operator_group* row in Table 4-1, these operators were derived from the methods permitted within the java.util.set interface[32]. These operator groups supported semantic attributes to be compounded into Semantic Attribute Queries, which was specific functionality outlined in Section 3.6.1.1 that needed to be supported.

The second method that the SARA API offers is *GetSemanticAttributes* which is called at design time by application developers wishing to know what semantic attributes are available in the installation of SARA to which they are connected. This API method is central to supporting the SARA design feature F3 as outlined in Section 4.2. A description of the *GetSemanticAttributes* method follows:

GetSemanticAttributes

public ArrayCollection *GetSemanticAttributes*(int *SemAttType*, int *SourceType*)

32 http://download.oracle.com/javase/1.4.2/docs/api/java/util/Set.html

Parameters:
SemAttType – Integer specifying the type of semantic attribute wanted:

```
     0 = All
     1 = Expert
     2 = Template
     3 = Hybrid
```

SourceType – Integer specifying the underlying source type wanted:

```
     0 = All
     1 = XML
     2 = RDF
     3 = Web Service
```

Returns:
An ArrayCollection in the format of Table 4-2. If a field is not relevant in specific situations (e.g. template_rules, var_name and var_types in an expert semantic attribute) then an empty ArrayCollection is sent.

Table 4-2. Representation of an ArrayCollection Returned by GetSemanticAttributes API Method

Pos.	**Name**	**DataType**	**Description**
0	sem_att_name	String	The name of the semantic attribute.
1	sem_att_type	String	Type of semantic attribute (*Expert*, *template* or *hybrid*)
2	param_name	Array Collection	The parameter names
3	rules	Array Collection	Any *expert* rules (corresponds to parameters)
4	template_rules	Array Collection	Any *template* rules (corresponds to parameters)
5	var_names	Array Collection	Any variable names
6	var_types	Array Collection	Any variable data types (corresponds to the variable names)

The final API method is *LoadSemanticAttributes* and is called once by applications while launching in order to initialise SARA for their use. This method ensures all semantic attributes are loaded into memory.

LoadSemanticAttributes

```
public boolean LoadSemanticAttributes()
```

Parameters:
none

Returns:
true if semantic attributes are successfully loaded, *false* if not.

4.2.6 Parsing of Queries and Reconciliation of Results

When a Semantic Attribute Query is received by SARA through its API's *QuerySARA* method, it is passed to the Query Decomposer component which supports the SARA design feature F6 outlined in Section 4.2. This component groups the constituent semantic attributes according to the *operator_group* parameter in Table 4-1. Regardless of the *operator_group* to which the semantic attribute belongs, its query is initially sent to the source location. If it is a *template* semantic attribute that requires tailoring by the end user, or a *hybrid* semantic attribute that the user has decided to tailor, the content of the *tailored_args* ArrayCollection populates the query variables before it is sent to the source. Any result identifiers that match the query in the source are then returned to SARA and placed into a *HashSet* in the Result Reconciler component of the framework. This component is responsible for supporting the SARA design features F7 and F8 outlined in Section 4.2, and its processes are now described.

If a semantic attribute query has more than one semantic attribute in the *operator_group* "MustHaveAll" then the intersection of the result sets is first calculated, and if the *operator_group* "MustNotHaveAll" has more than one semantic attribute then the intersection of the result sets is also found first. Likewise, if *operator_group* "MustHaveAtLeastOne" has more than one semantic attribute then the union of these result sets is first calculated, and if *operator_group* "MustNotHaveAny" has more than one semantic attribute then the union of these result sets is also found first. This format allows great flexibility in the type of queries that the end user can form. For instance the following is an example of a complex Semantic Attribute Query containing nine semantic attributes (SA1 – SA9) in three *operator_groups*:

Return all Superclasses that are SA1, SA2 and SA3, that are either SA4 or SA5 or SA6 and that are not SA7 or SA8 or SA9.

After SARA parses this, it routes queries to the individual data sources and receives the results back for all nine separate semantic attributes:

- *SA1-SA3* belong to *operator_group* "MustHaveAll"; the intersection of their results form a new set *A*.

  $$SA1 \cap SA2 \cap SA3 = A$$
- *SA4-SA6* belong to *operator_group* "MustHaveAtLeastOne"; the union of their results form a new set *B*.

  $$SA4 \cup SA5 \cup SA6 = B$$
- *SA7-SA9* belong to *operator_group* "MustNotHaveAny"; the union of their results form a new set *C*.

  $$SA7 \cup SA8 \cup SA9 = C$$
- Sets A and B are then intersected, followed by an asymmetric set difference between the newly intersected set and *C*.

  $$(A \cap B) / C = \textit{Result Set}$$
- This produces a final result set of identifiers which get constructed into an XML Result Model and sent back to the client application for rendering.

This query presumes that *SA1-SA9* are all associated with the same superclass. However as described in Section 3.6.3, if a user is to make a query that spans multiple superclasses such as "Return all *Albums* that contain UK number 1 *Songs* by *MusicArtists* that have played Ireland in the last year", a mechanism for superclass transformation is necessary before result reconciliation can take place. Because this query returns *Albums*, all other superclasses in this query must have their results transformed into *Albums* also. SARA supports this automatically by using the superclass conversion data stored in the Domain Superclass Model.

When the result sets are sent back to SARA from each constituent semantic attribute's source they are first checked to see if they are associated with the return superclass. If any of the result sets are associated with a different superclass to that specified in the Semantic Attribute Query, then a conversion of these result sets is needed and an XQuery over the Domain Superclass Model is executed at run time. Figure 4-11 shows sample code from the Domain Superclass Model which would be accessed by this XQuery.

```
1       <Superclass>
2         <Name>Music_Artist</Name>
3         <Conversion>
4             <ConversionInput>Album</ConversionInput>
5             <ConversionAddress>http://virtuoso.dbtune.org/sparql
6             </ConversionAddress>
7             <ConversionQuery>
8             PREFIX mysp:<http://purl.org/ontology/myspace>
9             PREFIX foaf:<http://xmlns.com/foaf/0.1/>
10            PREFIX mo:<http://purl.org/ontology/mo/>
11            SELECT DISTINCT ?id
12            FROM <http://dbtune.org/myspace/>
13            WHERE
14              {
15                ?result mysp:name ?id.
16                ?album mysp:hasArtist ?result.
17                ?album mysp:title ?1 .
18                FILTER (?1 = " + Variable_1 + ").
19              }
20            </ConversionQuery>
21            <ConversionFormat>RDF</ConversionFormat>
22            <ConversionRationale>MySpace has a comprehensive list of
23            official and unofficially released albums by artists in
24            musicbrainz format
25            </ConversionRationale>
26            <Priority>1</Priority>
27        </Conversion>
28    </Superclass>
```

Figure 4-11. Sample Domain Superclass Model

If a conversion from *Albums* to *Music_Artists* is required, then the first step of this XQuery is to locate the superclass that the query will return (line 2) and identify the superclasses that can convert to it directly (line 4). If the superclass that needs to be converted (*Album*) is listed as one of the ConversionInputs then the corresponding query (lines 7-20) is extracted and the set of *Album* identifiers that need converting are slotted into the variable position in the query (line 18). These queries are then fired off to the transformation source (line 5) with the result set of superclass instances (*Music_Artists*) loaded into a HashSet in SARA. The type of superclass dictates whether there will be more or fewer results than in the original result set before its conversion e.g. there are typically less *MusicArtists* than *Albums*. The use of a HashSet at this stage also ensures that there are no duplicates in the result sets.

If there is no direct conversion available in the Domain Superclass Model then the XQuery recursively checks to see if a multi-step conversion can take place. For instance this would occur if a query necessitated conversion from *Songs* to *Venues* but *Venues* only could convert to *Countries* and *MusicArtists*. The XQuery automatically detects if *Countries* or *MusicArtists* can be directly converted to *Songs*, and if so triggers a conversion of all the *Songs* to *MusicArtists* and then all the returned *MusicArtists* into *Venues*. The XQuery will recursively check the entire Domain Superclass Model to find the most direct route for conversion, and if there are multiple conversion options for single steps, the higher priority conversion (line 26) determines which gets chosen. If the superclass conversion has to be done by a service behind a native API rather than a source directly accessible by a query language, the related wrapper in SARA acts as a middleman. Just like the Domain Superclass Model itself, this conversion functionality offered by the wrapper is reusable in different SARA installations.

The superclass conversion functionality within SARA requires that a minimum of one further query be sent for each result (i.e. sending a query to convert from an Album ID to an Artist ID). Hence, if there is a large amount of results from the original query sent by the client application, then this initial query may spawn many hundreds or thousands of extra queries. The speed of these queries is very dependent on the particular data source being accessed and the network conditions at that time. The current prototype of SARA sends these extra queries one at a time, with the next query firing after the results from the previous query have been returned. This was as a result of a decision to focus on implementing the core functionality of SARA rather than ensuring that all its operations would perform to a production level performance. Hence, the SARA prototype is only suitable for handling superclass conversions if there are a small amount of results, and the application does not view these queries as time critical.

4.2.7 Summary and Analysis

This section has described the implementation of SARA and how it satisfies each of the requirements of the Reconciliation Engine component that were outlined in Chapter 3. The models supported by SARA, the underlying technologies that enable it and the various interfaces that facilitate its communication have been described in detail. In particular it was shown how the implementation embraces a modular approach, with particular care given to ensuring that SARA supports the reusability and extension of models, as well remaining independent of any particular domain. Furthermore, the implementation of SARA showed how multiple sources of different data formats can co-exist, and that results can be reconciled for queries spanning more than one data source. The implementation of SARA mirrors closely the design requirements set out for it in Chapter 3, hence the evaluation of SARA described in the following chapter will also give a good indicator as to the success of the implementation's underlying design and approach. The next section will describe how SABer supports non-technical domain experts to create semantic attributes, and how it works in tandem with SARA to make these semantic attributes available to client applications and their users.

4.3 SABer (Semantic Attribute Builder)

One of the primary goals of the research described in this thesis is to enable non-technical domain experts to encode subject matter expertise (SME). As described in Chapter 3, this approach requires SME to be encoded in the Semantic Attribute Model. Hence an authoring tool is needed that works in tandem with SARA to help technical and non-technical domain experts to generate semantic attributes. The requirements for such a tool as outlined in Section 3.8 are as follows:

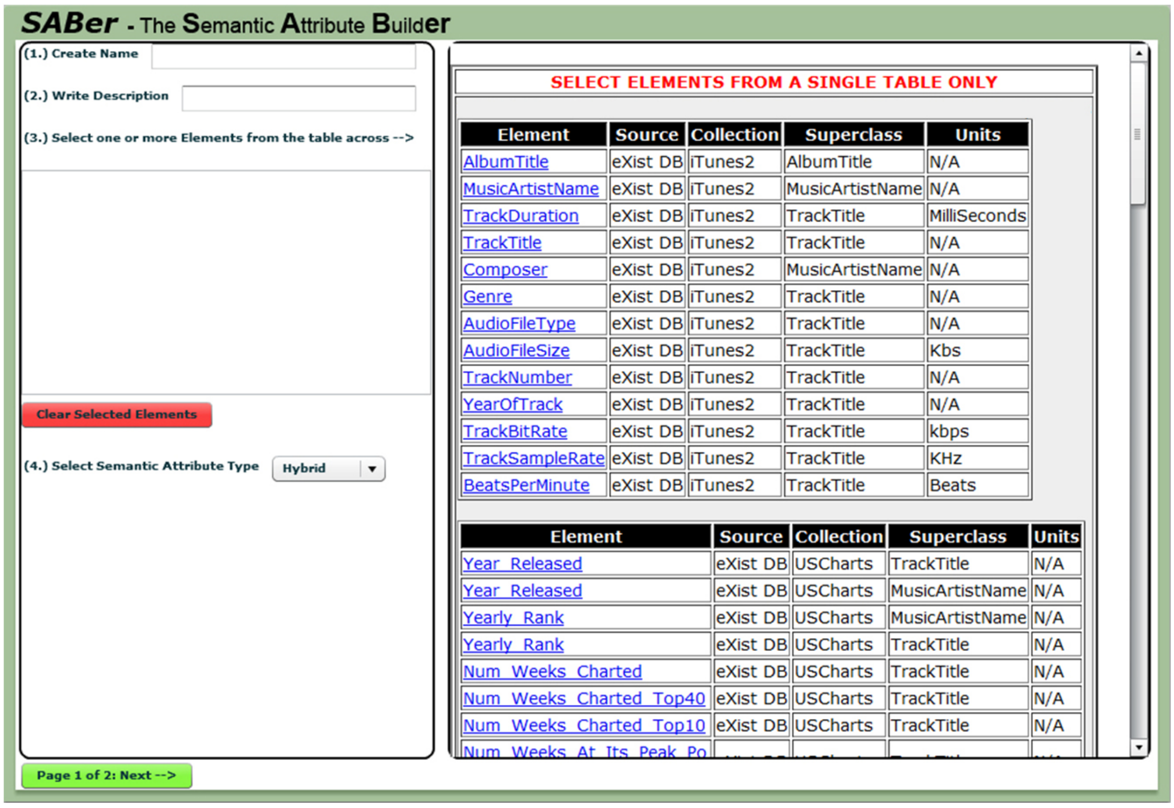
- Be accessible to users with no computer coding or information modelling background, after minimal training
- Automate as much of the creation process as possible to support these users
- Support a schema-based approach to query building using a wizard/form interface
- Work in tandem with the Reconciliation Engine.
- Incorporate the Source Registry to display available metadata
- Group this metadata and make available for rule generation.
- Support rule generation for semantic attributes in multiple query languages.
- Be extensible for new data formats.
- Be able to query multiple data sources for the results to rules being generated, so as to provide instant feedback to domain experts as what the end user is to expect.
- Allow all three semantic attribute types to be formed.
- Generate the Semantic Attribute Model as an output file.

The Semantic Attribute Builder (SABer) was developed in Adobe Flex to satisfy these requirements and deployed as an Adobe Air desktop application. Its main aim is to allow non-technical users to encode their expertise in SPARQL, XQuery or as native API calls, and to encapsulate this SME in an XML Semantic Attribute Model. It achieves this by automating as many processes as possible, ensuring that the rules generated are syntactically correct and do not require the domain expert to understand XML or the underlying query languages. Any Semantic Attribute Models exported by SABer get imported into SARA's Semantic Attribute Library so that client applications, and by extension end users, can gain access to them.

Creating a semantic attribute using SABer is a two step process with each step having a dedicated page in the application. The first step is to name and describe the semantic attribute, and then to select its type and its component metadata. In the second step, domain experts use the metadata they have selected from step one to generate one or more rules for the semantic attribute. These rules are formed via a schema-based approach to query building which was identified in chapter two as having most potential for non-technical domain experts to encode SME. Each of the two steps and how they are implemented within SABer are now described in detail, and are followed by a section on SABer's interaction with SARA.

4.3.1 Semantic Attribute Authoring Process in SABer - Step One

The first process in step one is to name the semantic attribute being created. Each name given to a semantic attribute must be unique to that installation of SARA, and it is important to choose a descriptive name that conveys its meaning clearly. It is this name that end users will see in the client application; hence it is important they have a clear idea of what they are selecting. A longer description with exact details of the source can be included in SABer's additional description field so that client applications will be able to unambiguously describe what each semantic attribute is conveying. Both these fields are free text, though SABer does limit the character set that can be inputted. Figure 4-12 shows how the SABer interface looks during step one.



SABer - The Semantic Attribute Builder

(1.) Create Name

(2.) Write Description

(3.) Select one or more Elements from the table across -->

(4.) Select Semantic Attribute Type: Hybrid

SELECT ELEMENTS FROM A SINGLE TABLE ONLY

Element	Source	Collection	Superclass	Units
AlbumTitle	eXist DB	iTunes2	AlbumTitle	N/A
MusicArtistName	eXist DB	iTunes2	MusicArtistName	N/A
TrackDuration	eXist DB	iTunes2	TrackTitle	MilliSeconds
TrackTitle	eXist DB	iTunes2	TrackTitle	N/A
Composer	eXist DB	iTunes2	MusicArtistName	N/A
Genre	eXist DB	iTunes2	TrackTitle	N/A
AudioFileType	eXist DB	iTunes2	TrackTitle	N/A
AudioFileSize	eXist DB	iTunes2	TrackTitle	Kbs
TrackNumber	eXist DB	iTunes2	TrackTitle	N/A
YearOfTrack	eXist DB	iTunes2	TrackTitle	N/A
TrackBitRate	eXist DB	iTunes2	TrackTitle	kbps
TrackSampleRate	eXist DB	iTunes2	TrackTitle	KHz
BeatsPerMinute	eXist DB	iTunes2	TrackTitle	Beats

Element	Source	Collection	Superclass	Units
Year_Released	eXist DB	USCharts	TrackTitle	N/A
Year_Released	eXist DB	USCharts	MusicArtistName	N/A
Yearly_Rank	eXist DB	USCharts	MusicArtistName	N/A
Yearly_Rank	eXist DB	USCharts	TrackTitle	N/A
Num_Weeks_Charted	eXist DB	USCharts	TrackTitle	N/A
Num_Weeks_Charted_Top40	eXist DB	USCharts	TrackTitle	N/A
Num_Weeks_Charted_Top10	eXist DB	USCharts	TrackTitle	N/A
Num_Weeks_At_Its_Peak_Po				

Figure 4-12. SABer Interface During Step One

The next task to complete on this page is the selection of metadata from which the semantic attribute rules will be created. Each semantic attribute requires a minimum of one metadata element. Within SABer the metadata is visualised on the right side of the screen after an XSL transform[33] of the Source Models in SARA which are in XML format. The resultant HTML version of these models are then rendered within the application and grouped by source data type. The current version of SABer supports three data types, XML, RDF and data accessible through a native API. It should be noted that the data stored in each individual Source Model plays a central role in defining the scope of SME that users can generate within SABer.

Each metadata element is rendered as a HTML link and can be selected to be part of a semantic attribute just by clicking on it. Apart from the element's name (typically displaying its alias description from the Source Model rather than the element name in order to reduce ambiguity), additional data from the Source Models such as its source data type, units and superclass are also displayed in SABer. Chosen metadata elements selected in error can be removed by pressing the "Clear Selected Elements" button and new ones

[33] http://www.w3.org/TR/xslt20/

can be selected in their place. There are no limits to the amount of metadata elements a person can choose for a semantic attribute, and there is no obligation to use every element chosen to form rules in step two of the process. The only exception is that data stored behind an API can only have one of its methods (represented as a single metadata element within SABer) used per semantic attribute. However, as mentioned in the design chapter, there is no limitation on two semantic attributes that employ different methods from a single web service API being combined into a Semantic Attribute Query within the client application.

There are other restrictions as to what metadata can be joined together into a single semantic attribute, hence SABer performs checks on all the metadata selected and displays warning messages if an incorrect selection has been made e.g. selecting metadata from two different sources or from two separate collections within a single XML database. Once the domain expert is satisfied with the metadata that he has chosen, he must select from a drop down menu the type of semantic attribute they want to create. Domain experts have a choice of three; *expert, template* or *hybrid.* As described in Section 3.6.1 an *expert* semantic attribute only contains the *expert's* default rule(s) which can't be tailored, a *template* semantic attribute contains no *expert* default rule(s) and must be tailored by the end user, and a *hybrid* semantic attribute contains *expert* default rules as well as corresponding *template* rules which can be tailored. When the user is satisfied with his choices he can click to move onto the next stage. A user can return to step one and make any adjustments before returning back to step two at any time. However, before proceeding to the next step, SABer performs checks to ensure that the user has selected at least one metadata element with which to create some rules.

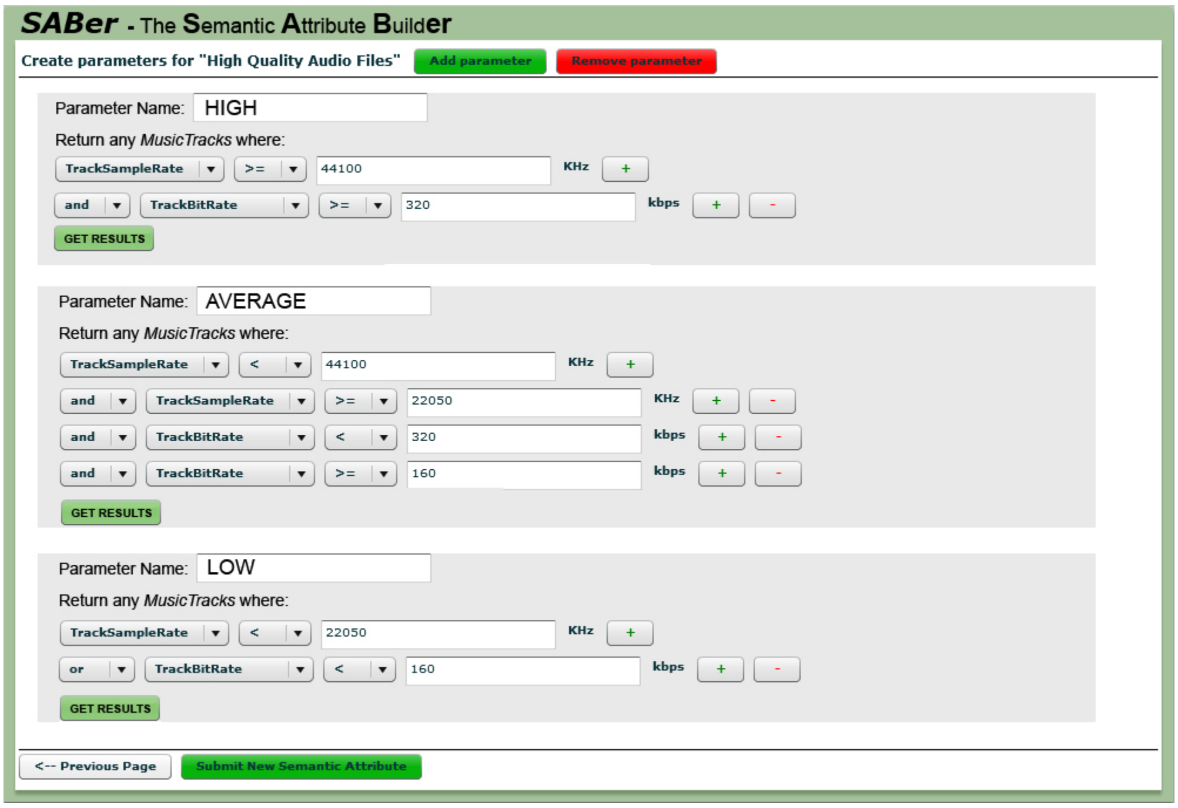
4.3.2 Semantic Attribute Authoring Process Step Two

Depending on whether the domain expert has selected an expert, *template* or *hybrid* rule the next page displayed will vary slightly. However, regardless of the type of semantic attribute being created, it is at this stage that the domain expert creates the rule or rules for their semantic attribute. Table 4-3 summarises how the various data types have rules generated for the different semantic attribute types.

Table 4-3. Summary of how Data Types have Rules Generated for Different Semantic Attributes

	Expert	**Template**	**Hybrid**
XML	Generate rules in constrained XQuery with specific values.	Generate rules in constrained XQuery but with no specific values	Identical to *expert* semantic attribute rule creation except corresponding *template* rules are auto-generated.
RDF	Generate rules in constrained SPARQL with specific values.	Generate rules in constrained SPARQL though with no specific values	Identical to *expert* semantic attribute rule creation except corresponding *template* rules are auto-generated.
API	The API parameters must be filled in text fields. Depending on web service wrapper, extra rules may be generated with specific values.	The API parameters for this type of semantic attribute rule are automatically filled in by SABer. User can only submit it.	Identical to *expert* semantic attribute rule creation except corresponding *template* rules are auto-generated.

Figure 4-13 shows what the SABer overall interface looks like during step two of the creation of a *hybrid* XML based semantic attribute. The next sections will describe in detail how *expert*, *template* and *hybrid* semantic attributes are generated in SARA for XML sources, RDF sources and data sources behind a native API.



SABer - The Semantic Attribute Builder
Add parameter
Remove parameter
GET RESULTS
<-- Previous Page
Submit New Semantic Attribute

Create parameters for "High Quality Audio Files"

Parameter Name: HIGH
Return any *MusicTracks* where:
TrackSampleRate >= 44100 KHz
and TrackBitRate >= 320 kbps

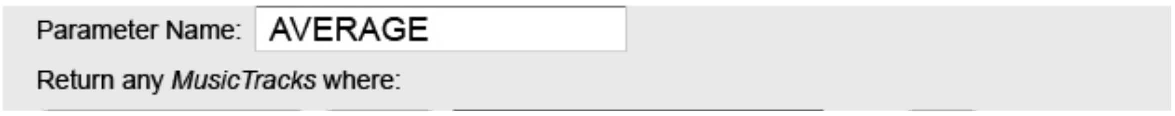
Parameter Name: AVERAGE
Return any *MusicTracks* where:
TrackSampleRate < 44100 KHz
and TrackSampleRate >= 22050 KHz
and TrackBitRate < 320 kbps
and TrackBitRate >= 160 kbps

Parameter Name: LOW
Return any *MusicTracks* where:
TrackSampleRate < 22050 KHz
or TrackBitRate < 160 kbps

Figure 4-13. SABer Interface During Step Two

4.3.2.1 Creating rules for an XML based Semantic Attribute

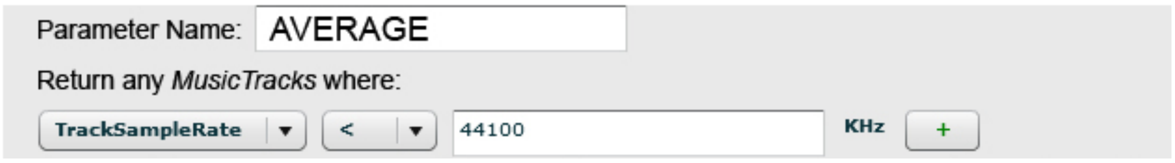
The process for creating rules for XML based semantic attributes is the same whether building an *expert* or *hybrid* semantic attribute. Each semantic attribute must have a minimum of one parameter and a domain expert may add and remove further parameters which have unique names. Once a parameter has been named, the domain expert can then start to generate an XQuery rule for it. The first part of each parameter's rule will have already been generated by SABer and is printed onscreen as "Return all <*Superclasses*> where" with the actual superclass of the metadata they have chosen in step one printed in place of <*Superclasses*>. Thus in the music domain the expert may be presented with "Return any *Albums* where", or "Return any *MusicTracks* where" as the start of their parameter's rule. Figure 4-14 depicts a parameter named *AVERAGE* and the first line of its rule which is automatically generated by SABer.



Parameter Name: AVERAGE
Return any *MusicTracks* where:

Figure 4-14. Sample First Line of Expert Rule for XML Based Semantic Attribute in SABer

Underneath this line the expert is presented with two dropdown menus, a text field and a button aligned horizontally. The first dropdown menu contains all the elements (typically their aliases to reduce ambiguity) that they chose in the first step, as described in the previous section. Hence there could be just a single metadata element, or else there may be several. Figure 4-15 depicts the first expert generated line of an XML based semantic attribute. The user would select the metadata element in which they were interested and then move on to the second dropdown menu adjacent to it.

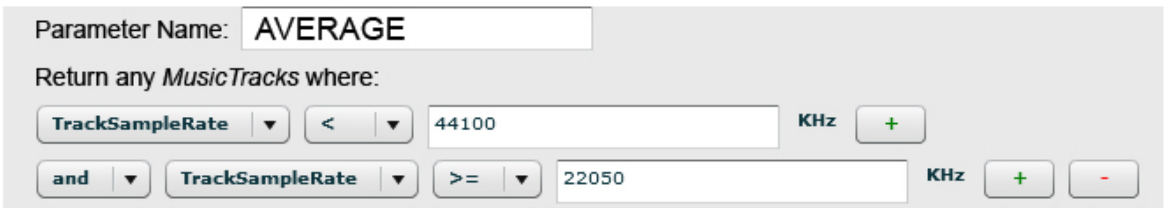


Parameter Name: AVERAGE
Return any *MusicTracks* where:
TrackSampleRate < 44100 KHz +

Figure 4-15. Sample Two Lines of Expert Rule for XML Based Semantic Attribute

This dropdown menu contains the available list of operators from which the end user can select from. Currently these are *Greater than, Less than, Equals to, Not Equals to, Greater than or Equals to, Less than or Equals to* and *Contains*. The domain expert simply selects which operator they want from the drop down menu. These operators enable experts to quantise domain properties into ranges, and are sufficient for initial experimentation. This is because it is not necessary for an individual semantic attribute to be overly complex (containing multiple clauses and operators), as much of SARA’s power is derived from Semantic Attribute Queries, which join these simple semantic attributes into complex compound queries.

All the domain expert has to do to finish this line of the rule is to input a value into the adjacent textbox. Thus in the Figure 4-15 the expert chose the metadata “TrackSampleRate”, the operator “<” and inputted the value “44100”. If there are any units associated with the metadata element in the Source Model (in this case KHz), they are automatically displayed at the end of the line in order to help the domain expert input appropriate values. If the domain expert wants to add more lines to this rule all they have to do is click on the “+” button at the end of the line. This adds another identical line underneath the first, except that it has an additional “and/or” dropdown menu at the start of the line and an additional “-” button at the end. Figure 4-16 displays an example of the first three expert generated lines.


Parameter Name: AVERAGE

Return any *MusicTracks* where:

TrackSampleRate < 44100 KHz +

and TrackSampleRate >= 22050 KHz + -

Figure 4-16. Sample Three Lines of Expert Rule for XML Based Semantic Attribute

The “and/or” dropdown menu allows the user to specify if the *MusicTracks* should satisfy all or either of the rules. In Figure 4-16 the expert has used “and” so only wants *MusicTracks* that satisfy both rule lines e.g. *MusicTracks* that have a sample rate less than 44,100Khz AND greater than or equal to 22,050KHz. The additional “-” button at the end of the line allows for a rule line to be deleted easily. Each parameter can contain as many rule lines as the expert wants. Figure 4-17 shows the completed five line rule. This essentially equates to the WHERE part of an XQuery statement with the rest of the XQuery automatically generated from the information defined in the Source Model. At any time in the process the domain expert can select the “Get Results” button to see what instances are currently in the data source that satisfy the rule being generated.

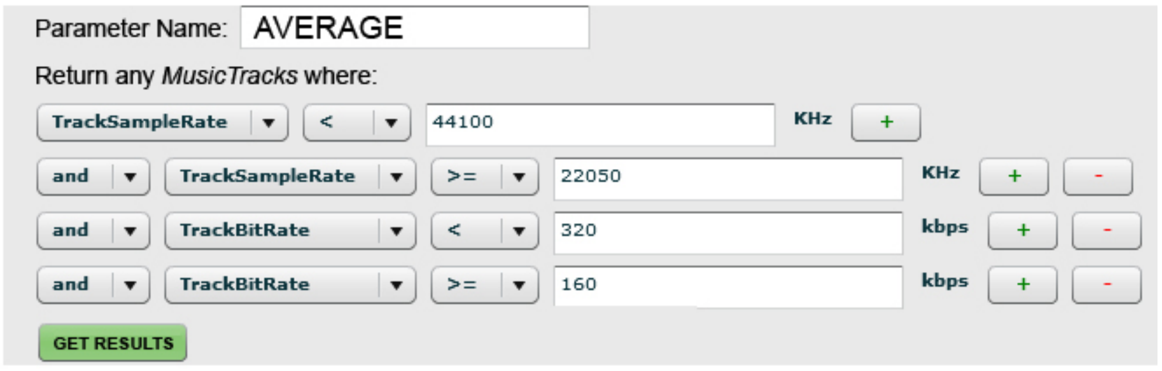


Figure 4-17. Sample Five Lines of Expert Rule for XML Based Semantic Attribute

If a semantic attribute only has a single parameter the domain expert can simply submit it once they are satisfied with the rule. However, if the semantic attribute requires multiple parameters he just clicks the "add parameter" button and repeats the process described above, submitting the semantic attribute when finished. As mentioned previously, the process of creating *expert* and *hybrid* semantic attributes for XML sources is identical. The only difference is that when a *hybrid* semantic attribute is submitted, SABer automatically generates a *template* rule for each of the *expert* rules and appends it to the Semantic Attribute Model.

The process for creating a *template* semantic attribute based on XML data is almost identical to the process just described for creating *expert* and *hybrid* semantic attributes. The only difference is highlighted in Figure 4-18.

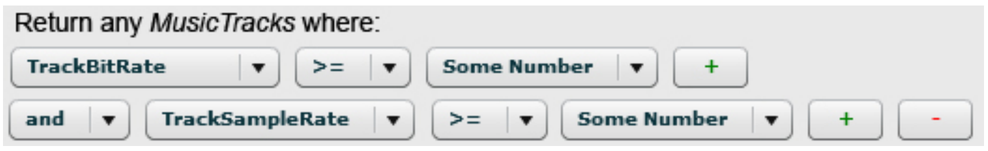


Figure 4-18. Sample Three Lines of Template Rule for XML Based Semantic Attribute

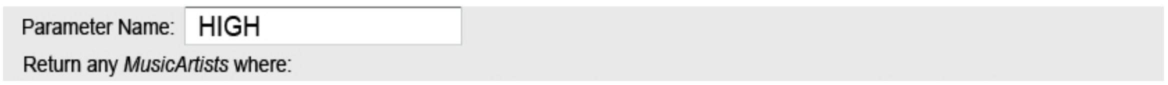
Instead of having a blank text field in which domain experts can input a specific value, they instead are presented with another dropdown menu with two wildcard options; "Some Text" and "Some Number". This allows domain experts to create rules such as *Return any MusicTracks where Artist Name = "Some Text"* or *Return any MusicTracks where chart position < "Some Number"*. Such rules enable end users to replace the wildcard options with explicit values, in order to tailor the rule more specifically to what they want.

4.3.2.2 Creating rules for an RDF based Semantic Attribute

The process of creating rules for RDF based semantic attributes is the same whether building an *expert* or *hybrid* semantic attribute. Each semantic attribute must have a minimum of one parameter and a domain expert may add and remove further parameters which have unique names. The SPARQL queries generated by SABer are more sophisticated than the XQuery generated because SPARQL allows *joins* to be expressed implicitly simply by including two *triple patterns* that reference a common variable. This feature enables individual *expert* rules in SPARQL to reference multiple Domain Superclasses.

The first thing a domain expert must do to generate their SPARQL query is to select the domain superclass that they want to return. Unlike in XML based sources, one is not tied to returning the same superclass as the metadata selected in step one. In fact, the domain expert has the choice of returning any of the superclasses referenced in the Source Model for that specific RDF source. To choose a superclass, the domain expert must simply select it from a dropdown menu at the top of the screen. This superclass will apply to all parameters for this particular semantic attribute.

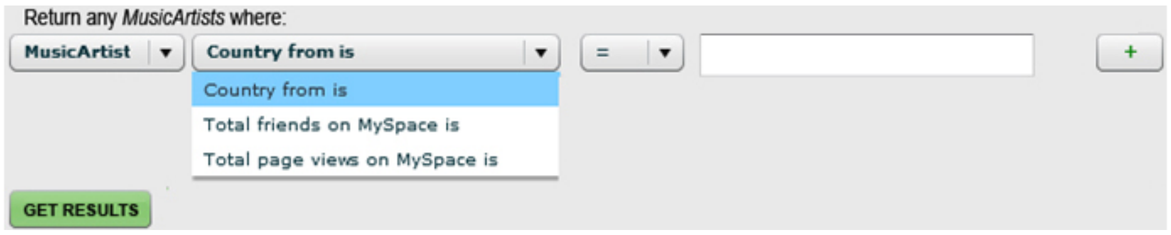
Once the superclass to be returned has been selected and a parameter has been named, the domain expert can then start to generate a SPARQL rule for it. The first part of each parameter's rule will already be generated by SABer and is printed onscreen as "Return any <*Superclasses*> where" with the actual superclass chosen previously from the dropdown menu in place of <*Superclasses*>. Thus in the music domain the expert may be presented with "Return any *Albums* where", or "Return any MusicArtists where" as the start of their parameter's rule. Figure 4-19 shows such a situation.


Parameter Name: HIGH

Return any *MusicArtists* where:

Figure 4-19. Sample First Line of Expert Rule for RDF Based Semantic Attribute

Underneath this line the expert is presented with three dropdown menus, a text field and a button, in a row. The first dropdown menu contains all the superclasses with which the RDF source has associated. Depending on what superclass the expert chooses, the predicates (or more precisely the alias of the predicates) in the adjacent dropdown menu will change accordingly. This second dropdown menu contains all the predicates selected in step one (see Section 4.3.1), but restricted to those that are associated with the superclass chosen in the first dropdown menu (these restrictions are specified in each Source Model). Thus Figure 4-20 shows that when the domain expert chooses the superclass *MusicArtist* from the first dropdown menu, he is presented in the second dropdown menu, with the elements *Country from is, Total Friends on MySpace is, and Total page views on MySpace*. Alternatively, if the domain expert had chosen *Song* as the superclass in the first dropdown menu, then the second dropdown menu would have been populated with *Track Duration*, *Composer*, *and Genre* etc. If there are any units associated with the metadata element, they are displayed at the end of the line to make clear to the domain expert what range of values is appropriate to input.



Return any *MusicArtists* where:
MusicArtist
Country from is
=
+
Country from is
Total friends on MySpace is
Total page views on MySpace is
GET RESULTS

Figure 4-20. Sample Two Lines of Expert Rule for RDF Based Semantic Attribute

The domain expert can then select the predicate he is interested in and move on to the third dropdown menu. Identical to how SABer deals with XML sources, this dropdown menu contains the available list of operators that the end user can select from. Currently these are *Greater than*, *Less than*, *Equals to*, *Not Equals to*, *Greater than or Equals to* and *Less than or Equals to*. The domain expert simply selects which operator they want from the drop down menu. The operators other than “Equals to” all result in a FILTER statement being added to the SPARQL rule that is in the process of generation. These operators enable experts to quantise domain properties into ranges, and are sufficient for initial experimentation. This is because it is not necessary for an individual semantic attribute to be overly complex (containing multiple clauses and operators), as much of SARA’s power is derived from Semantic Attribute Queries, which join these simple semantic attributes into complex compound queries.

All the domain expert has to do to finish this line of the rule is to input a value into the adjacent textbox. Thus in the Figure 4-21 the expert chose the metadata “Total Friends on My Space”, the operator “<” and inputted the value “50000”. This essentially equates to the WHERE part of a SELECT SPARQL statement with the rest of the query automatically generated from the information defined in the Source Model. Identical to how SABer deals with XML sources, if the domain expert wants to add more lines to this rule all they have to do is click on the "+" button at the end of the line. This adds another identical line underneath the first, except that it has an additional "and/or" dropdown menu at the start of the line and an additional "-" button at the end.

The "and/or" dropdown menu allows the user to specify if the *MusicArtists* should satisfy both of the rule lines or either of them. If the user selects "or" from this dropdown menu, it results in an OPTIONAL statement being added to the SPARQL rule that is being generated. In Figure 4-21 the expert has used "and" so only wants *Music Artists* that satisfy both rule lines e.g. Music Artists whose Number Friends on MySpace is less than 50,000 AND greater than 22,000. The additional "-" button at the end of the line allows for a rule line to be deleted easily. Each parameter can contain as many rule lines as the domain expert likes, with Figure 4-21 showing the completed four line rule for "Averagely Popular Irish Artists on MySpace". At any time in the process the domain experts can select the "Get Results" button to see what instances are currently in the data source that satisfies their rule.

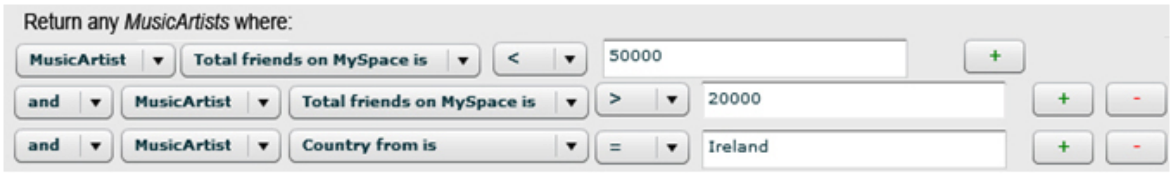
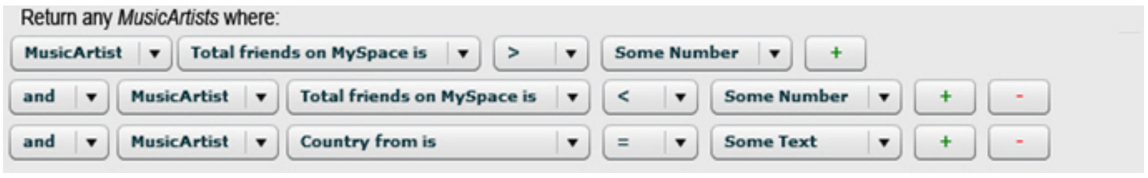

Return any *MusicArtists* where:
MusicArtist
Total friends on MySpace is
<
50000
+
and
MusicArtist
Total friends on MySpace is
>
20000
+
-
and
MusicArtist
Country from is
=
Ireland
+
-

Figure 4-21. Sample Four Lines of Expert Rule for RDF Based Semantic Attribute

If a semantic attribute only has a single parameter the domain expert can simply submit it, once they are satisfied with the rule. However, if the semantic attribute requires multiple parameters they just click the "add parameter" button and repeat the process described above, submitting the semantic attribute when finished. As mentioned previously, the process for a domain expert creating *expert* and *hybrid* semantic attributes for RDF sources is identical. The only difference is that when a *hybrid* semantic attribute is submitted, SABer automatically generates *template* rules for each of the *expert* rules and appends it to the Semantic Attribute Model.

The process for creating a *templat*e semantic attribute based on RDF data is almost identical to the process just described for creating *expert* and *hybrid* semantic attributes. The only difference is highlighted in Figure 4-22. Instead of having a blank text field in which domain experts can input a specific value, they instead are presented with another dropdown menu with two options "Some Text" and "Some Number". This allows domain experts to create rules such as *Return all Artists where Country From = "Some Text"* or

Return all Songs where chart position < "Some Number". By generating these kinds of rules it enables end users to tailor a rule more specifically to what they want.


Return any *MusicArtists* where:

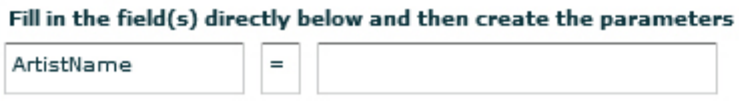
	MusicArtist	Total friends on MySpace is	>	Some Number	+	
and	MusicArtist	Total friends on MySpace is	<	Some Number	+	-
and	MusicArtist	Country from is	=	Some Text	+	-

Figure 4-22. Sample Four Lines of Template Rule for RDF Based Semantic Attribute

4.3.2.3 Generating Rules for Native API based semantic attributes

Creating rules for native API based semantic attributes is more restrictive than for those based on sources accessible by a query language. This is because APIs tend to require strict parameters to be sent in order to function correctly. However, there is still scope for experts to exploit these services to generate interesting rules, and the process for creating these rules is the same whether building an *expert* or *hybrid* semantic attribute.

Like XML based semantic attributes, a domain expert does not specify what superclass they want they return, as this has been predetermined by SABer from the relevant Source Model. The first thing that the domain expert must do is to fill in any arguments that the web service's API method requires. The expert is presented with the name of the argument and has a textbox to fill in what their answer. For instance Figure 4-23 shows that this particular API method requires one parameter called *ArtistName*.


Fill in the field(s) directly below and then create the parameters

ArtistName =

Figure 4-23. Sample API Parameter in Expert Rule for API Based Semantic Attribute

If a particular web service wrapper created for SARA only contains API arguments (see Section 3.6.2.3) for a particular API method, then once these have been filled in by the domain expert they can immediately submit the semantic attribute. However, if the web service has optional wrapper arguments defined in the Source Model then the domain expert must specify a minimum of one parameter with a unique name. SABer's interface makes clear to the expert whether this is necessary or not, so that they know what is expected of them in order to generate the semantic attribute.

Once a parameter has been named by the domain expert, he can then start to generate a rule for it. The first part of each parameter's rule will already be generated by SABer and is printed onscreen as "*<API method name>* where" with the name of the API method printed in place of "*<API method name>*. Figure 4-24 shows how a parameter for the *GetTopSongsByArtist* method appears in SABer. Underneath this line the expert is presented with the name of each parameter and a textbox for the value he wishes to specify. Unlike RDF and XML based semantic attributes they is no selection of operators necessary. In this instance the user might call the parameter *High* and specify *1* for *Highest_Popularity_Position* and *10* for *Lowest_Popularity_Position.*

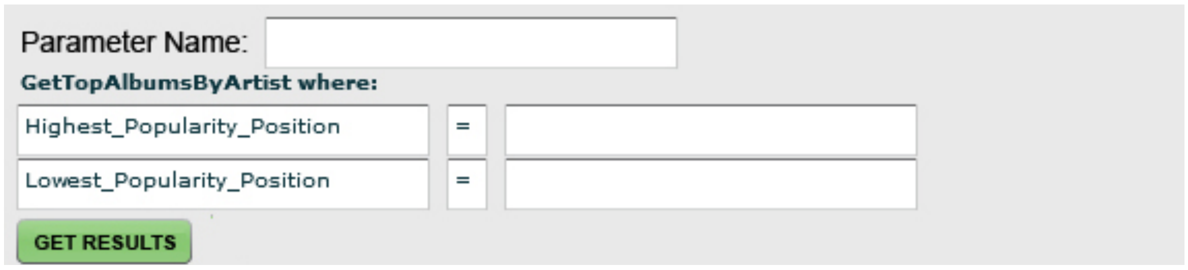
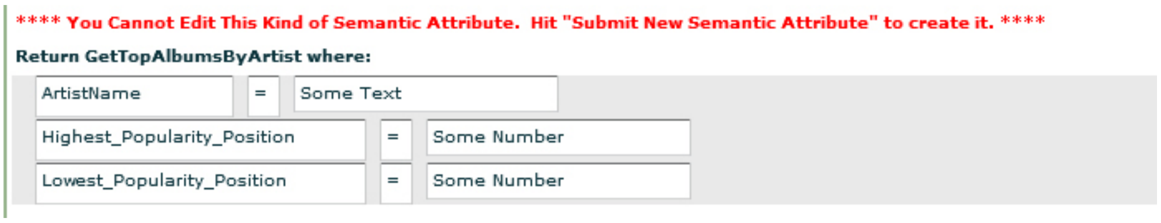

Parameter Name:
GetTopAlbumsByArtist where:
Highest_Popularity_Position
=
Lowest_Popularity_Position
=
GET RESULTS

Figure 4-24. Sample SARA Parameter in Expert Rule for API Based Semantic Attribute

At any time in the process the domain experts can select the "Get Results" button to see what instances are currently in the data source that satisfies their rule.

If a semantic attribute only has a single parameter the domain expert can simply submit it once they are satisfied with the rule. However, if the semantic attribute requires multiple parameters he just clicks the "add parameter" button and repeats the process described above, submitting the semantic attribute when finished. As mentioned previously, the process for a domain expert creating *expert* and *hybrid* semantic attributes for API accessible sources is identical. The only difference is that when a *hybrid* semantic attribute is submitted, SABer automatically generates *template* rules for each of the *expert* rules and appends it to the Semantic Attribute Model.

If a domain expert decides to generate a *templat*e semantic attribute from a web service then they cannot make any adjustments. The arguments required by the API method are automatically populated by SABer with variables, and the semantic attribute just needs to be submitted by the user. Figure 4-25 shows a *template* rule automatically generated for an API based semantic attribute.


**** You Cannot Edit This Kind of Semantic Attribute. Hit "Submit New Semantic Attribute" to create it. ****

Return GetTopAlbumsByArtist where:

ArtistName	=	Some Text
Highest_Popularity_Position	=	Some Number
Lowest_Popularity_Position	=	Some Number

Figure 4-25. Sample Template Rule for API Based Semantic Attribute

4.3.2.4 SABer Interactions with SARA

Once a semantic attribute has been submitted, the values and rules that have been inputted into SABer get concatenated with a template to form an XML Semantic Attribute Model (as described in Section 4.2.2.3). This model gets saved to a specific directory in the Semantic Attribute Library depending on the data type of its source. It then gets parsed by SARA and converted into its own Java method as described in Section 4.2.4. If the domain user presses the "Get Results" button while creating a rule, a stub semantic attribute XML model is generated in SABer and sent to SARA. There is a stub Semantic Attribute Model for each data format supported (currently XML, RDF and web services) and this model is parsed by JDOM to extract the query under test. In a similar process to that described in Section 4.2.3 (SARA's interface with client applications) the query gets sent to the relevant source, with the XML Result Model sent back to SABer via BlazeDS and rendered in a pop up window. This process is highlighted in the dashed lines of the architecture depicted in Figure 4-26. Because there is only ever one source per individual semantic attribute there is no need for any reconciliation of results to take place before sending back to SABer.

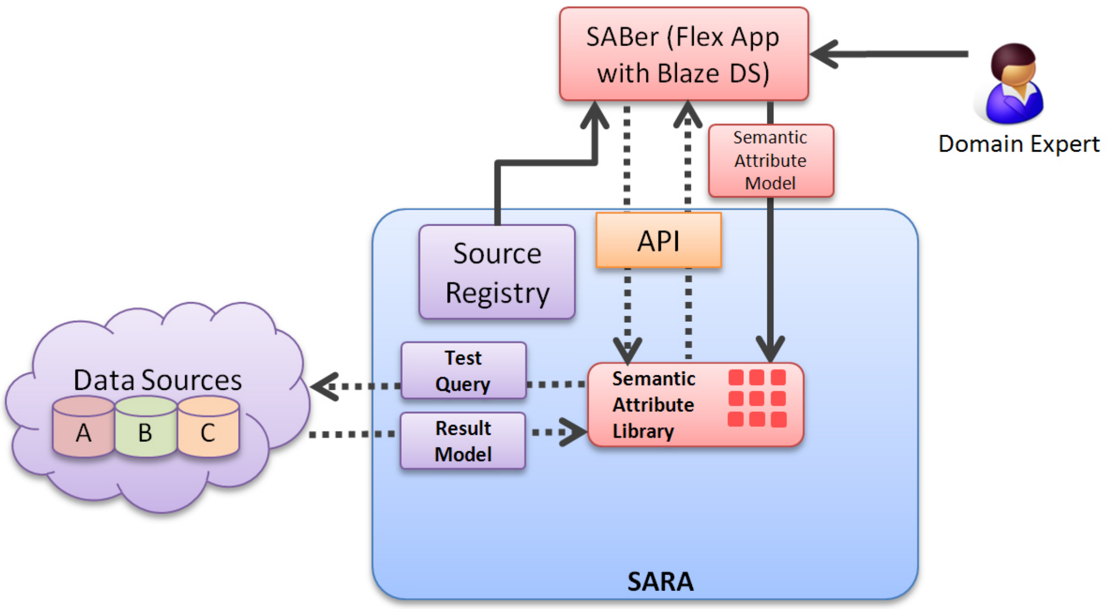


Figure 4-26. Architecture of SABer's Interaction with SARA

4.3.2.5 Summary and Analysis

This section discussed the implementation of SABer designed to support technical and non-technical domain experts in encoding SME in the Semantic Attribute Model. This two step process has been described for all types of semantic attributes (*expert, template* and *hybrid*) and the three data types that SABer supports (XML, RDF and API based). The way in which SABer works in tandem with SARA to show results while rules are being generated, as well as how the submitted semantic attributes are available to client applications through SARA has also been described. The implementation of SABer has been shown to satisfy each of the requirements for the SME Authoring Tool outlined in Chapter 3, and the features it employs to support non-technical domain experts in encoding SME have been discussed in detail. The evaluation of SABer in Chapter 5 will highlight the usefulness and usability of the system, and because its implementation closely mirrors the design requirements, the evaluation of SABer will also be a good indicator as to the usefulness of the underlying design. The next section will briefly describe a case study showing the use of SARA and SABer in helping exploration of the music domain.

4.4 Music Domain Case Study

This section briefly describes a case study of a SARA installation that connects to a dataspace of five separate music data sources in three different formats. The data sources used in this case study are also employed in the user trials performed with SABer, which are discussed in Section 5.6 of the evaluation chapter which follows. However, it should be noted that SARA has been successfully implemented in a number of other domains, including digital imaging and films, which are also detailed in the evaluation chapter (Sections 5.5.2 and 5.5.5 respectively). The main aims of this particular case study are to show how the SARA implementation can:

- give client applications consolidated access to multiple sources of various data types.
- support values sent through its API to tailor the underlying rules inside the semantic attributes.
- enable data to reside in its original location, and support instance level reconciliation between the different data sources.

The following are the five sources used in this case study:

1. An iTunes library with over 30,000 songs stored in XML in an eXist database[34]
2. The entire US Singles charts from 1950-2008 stored as XML in an eXist database[35]
3. The freebase.com music SPARQL endpoint[36]
4. The MySpace.com SPARQL endpoint[37]
5. Last.fm web services[38]

Each of these five sources chosen had Source Models registered to the Source Registry which in turn were visualised in SABer. The Domain Superclass Model contained entries for *Artist, Song, Album* and *Country.* SABer was then used to create semantic attributes which were stored in SARA's Semantic Attribute Library. As will be shown in the Section 5.6, SABer can support non-technical domain experts to generate such semantic attributes. For this case study, twenty-five semantic attributes for the domain were created including:

- Artists currently touring specific countries
- Top MySpace artists from specific countries
- Popular Jazz artists in the US Charts in the 1980s
- Similar artists to a specified artist

All Semantic Attribute Models stored by SARA were parsed so that a corresponding Java method encapsulating the expert-generated rules was accessible at run time.

Once the semantic attributes are made available in SARA it was possible for Semantic Attribute Queries to be sent to SARA from a client application via its API. For instance, queries combining multiple semantic attributes that reference different sources could be sent to SARA such as:

- *Return all Music Artists from the iTunes collection* (iTunes XML database), *that have Concerts Scheduled in the USA* (Last.fm web service)*, despite their most recent top 10 Album in the USA being more than ten years ago* (US charts database).

- *Return all Countries* (MySpace SPARQL endpoint) *that had popular Artists in the USA during the 1990s* (US charts database).

[34] See Appendix B for sample XML
[35] See Appendix C for sample XML
[36] http://lod.openlinksw.com/sparql
[37]http://virtuoso.dbtune.org/sparql
[38] http://www.last.fm/api

- *Return all Songs by The Beatles* (freebase SPARQL endpoint) *that are in top 10 popular Beatles songs on Last.fm* (Last.fm web service) *despite not charting in the top 10 in Americas* (US charts database).

Many of these Semantic Attributes Queries allowed specific values to be tailored by the end user, so that they could easily specify different music artists other than *The Beatles,* tailor the definition of *popular*, or change the range of time.

As mentioned in Section 3.6.2.3, the eXist databases and remote SPARQL endpoints could be directly accessed by queries encapsulated in the semantic attributes. However, in the case of web services with a native API, such as the Last.fm service used in this case study, a Java wrapper was needed to proxy queries and results. Once the results from the individual data sources were sent back to SARA they were reconciled into a final result set. This set was then used to populate the XML Result Model returned to the client for rendering.

This case study has shown how SARA supports semantic attributes created in SABer to be utilised by a client application. This gives the user consolidated access to multiple sources of different types, and enables instance level integration of results from several data sources residing in their original location. Furthermore, the case study showed how SARA used values sent through its API to tailor the underlying rules inside semantic attributes. Finally, it should be noted that extra superclasses, data sources and semantic attributes could be appended to the system seamlessly if required, and that the models generated for this case study could be plugged into different installations of SARA concerning the music domain.

4.5 Extending SARA and SABer for new data formats

SARA and SABer initially worked purely with XML data sources, but were subsequently extended to support RDF and data accessible through Web APIs. The process of extending these systems for further data formats is as follows. First a new Source Model must be developed for the specific data format, which describes how the data can be accessed from sources of this type and queries sent to them. SARA must then be extended to be able to parse this Source Model and to send runtime queries to data sources of this format. Finally, SABer must add support for the encoding of rules in a query language used by the new data format, and for these queries to be encapsulated as SME within the current Semantic Attribute Model. It is vital for the new additions to the SABer user interface to be tested with non-technical users to ensure that they can intuitively encode SME in this format. Thus several revisions and simplifications may need to take place. However, the important thing to stress is that this manual effort only takes place once for each data format, and once implementation occurs, both SARA and SABer are capable of handling data from any number sources using this data format.

4.6 Summary

This chapter has described how the main framework components to support the *expert-supported approach to data exploration* (the Reconciliation Engine and SME Authoring Tool), were implemented as SARA and SABer. Each of these components and the interfaces and models necessary to support them were described in detail, as well as a case study in the music domain that highlighted some of the features SARA and SABer offer. The following chapter will detail the different ways that SARA and SABer were evaluated throughout the course of this thesis.

5 Evaluation

This chapter describes the overall evaluation strategy employed in this thesis, as well as detailing the various experiments involved. These evaluation studies incorporated a variety of techniques, such as user trials, performance tests, questionnaires and interviews, and included experiments with SABer, SARA and the third party applications that used them. This chapter concludes with an analysis of the evaluation results, a brief comparison with the state of the art, and an overall summary.

5.1 Introduction and Evaluation Overview

The research objectives for this thesis (specified in Section 1.3), which were derived from the research question are as follows:

1) Analyse the state of the art in data exploration to determine the extent to which casual users are facilitated, and examine the state of the art in SME encoding for non-technical experts to identify the main features of current approaches.
2) Define an approach that allows end users to leverage SME (tailoring as appropriate) when exploring and consolidating information from separate data sources in a domain, and describe the accompanying models and framework necessary to implement it.
3) Perform evaluation studies to assess:
    a) the usability (effectiveness, efficiency and satisfaction) (Jokela et al. 2003), of the implemented SME authoring component of the framework, and the ability of non-technical users to encode SME.
    b) whether the encoded SME can be usefully exploited by client applications to adequately support end-user exploration.
    c) the features of the framework implementation and whether the consolidation of data from separate sources is supported.

How Objective 1 was realised was discussed in Chapter 2, with a description on how Objective 2 was attained detailed in Chapters 3 and 4. This chapter focuses specifically on Objective 3, which consists of three distinct evaluation objectives. In order to increase the granularity of these evaluation objectives (specifically 3c), it was necessary to refine them into specific features that SARA or SABer should contain. The evaluation experiments described in this chapter are thus used to provide evidence that the following features are being supported:

E1 SARA provides client applications with access to data sources without prescribing a specific user interaction paradigm for the GUI, or for their developers to know the underlying query language associated with each source.

E2 SABer enables the SME leveraged by the client applications to be encoded by non-technical experts.

E3 Casual users that appropriate (or tailor) this SME within a client application can send complex queries to data sources via SARA.

E4 SARA and SABer are domain independent.

The description of SARA and SABer's implementation in the previous chapter was accompanied by a music case study (Section 4.4) that highlighted design strengths of SARA were technically realised. This case study highlighted how SARA occupies a distinct niche within its field because of the range of features that it offers. Specifically this case study provided evidence that SARA:

- gives client applications consolidated access to multiple sources of various data types.
- supports values sent through its API to tailor the underlying rules inside the semantic attributes.
- enables data to reside in its original location, and supports instance level reconciliation between the different data sources.

These features complement the evaluation features discussed in this chapter and can be encapsulated as:

C1 Technical features of SARA showcased in music case study.

Finally, as middleware systems can cause an extra bottleneck that client applications must pass data through, it is also prudent to test SARA's performance in order to help quantify to what degree it increases query latencies. This procedure can be summarised as:

P1 Performance evaluation of SARA to see under what circumstances it is a usable middleware.

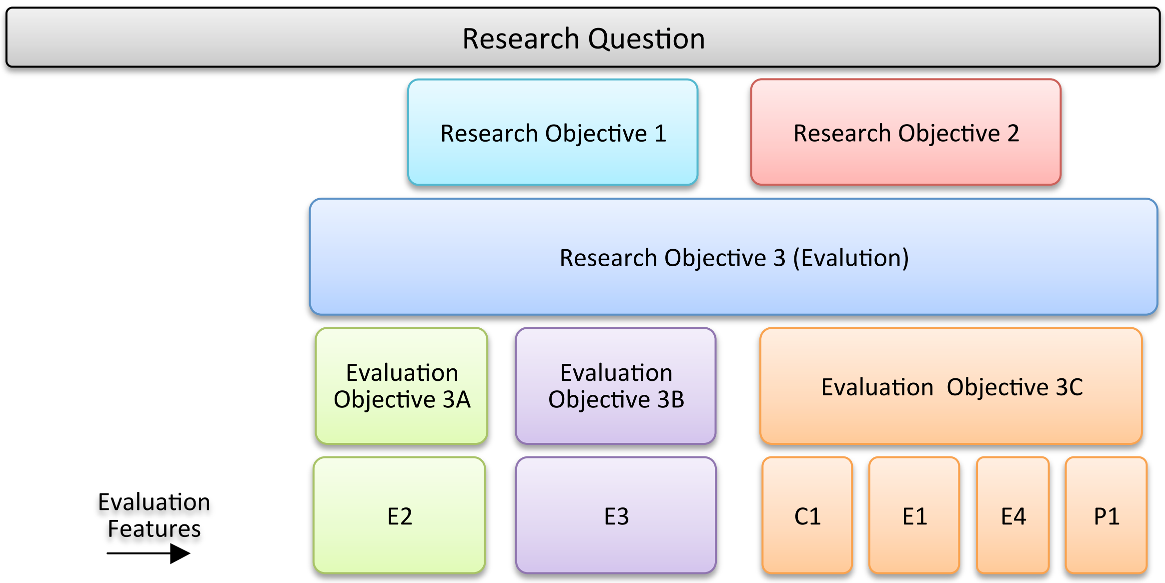


Figure 5-1. Thesis Hierarchy from Research Question to Evaluation Features

Figure 5-1 depicts the hierarchy that exists in this thesis in relation to evaluation (please note that the size of the boxes in the diagram is not representative of their relative importance). As is shown in the diagram, this hierarchy stretches from the research question described in Chapter 1, down to the individual features and procedures just summarised (E1-E4, C1 & P1). Verifying that the evaluation features (E1-E4) exist, involves several experiments using SARA, SABer and the client applications that use them. Furthermore, the performance evaluation of SARA (P1) also requires its own distinct experiment. As mentioned previously, the case study in Section 4.4 describes how SARA accommodates the features encapsulated in C1. Hence, this chapter does not reiterate these findings.

The experiments described in this chapter, and the evaluation findings presented, validate the research objectives of this thesis. Furthermore, because the implementation of SARA and SABer closely follow the framework design associated with the *expert-supported approach to data exploration*, their successful evaluation help provide validation of this approach and its associated framework.

5.2 Evaluation Strategy:

One of the main aims of this thesis' evaluation strategy was to get feedback from the key stakeholders at different stages, so that this information could feed directly back into the design process. Thus there was a need for initial user-centred feedback on the prototypes, as well as further evaluations on the refined systems. The stakeholders involved in these experiments included domain experts, application developers and end users of applications. In total there were five main experiments undertaken in this research, which used a number of different evaluation techniques such as structured interviews, questionnaires, performance tests and user trials. Each experiment (apart from Experiment 3) is discussed under three main headings; Experimental Goals, Experimental Setup and Experimental Results.

An initial experiment (Experiment 1) was needed to serve two main purposes. Firstly, it was necessary to examine if the approach implemented in the SARA prototype, where it acted as a mediator between a client application and multiple music data sources, was appropriate from a technical perspective. Secondly it was crucial to get some early qualitative feedback from end users as to whether SARA facilitated useful functionality that users of a client application could exploit. Hence, this experiment examined whether features E1 and E3 described in Section 5.1 were being supported by SARA. Furthermore, by gathering this information early in the development process it meant that user feedback could be quickly implemented into future iterations of the design.

As Experiment 1 focussed purely on SARA, it was prudent to perform evaluation tests with an early prototype of SABer also. Furthermore, as SARA relies on developers to create client applications that interface with it, it was useful to gather qualitative feedback on SARA from their perspective. In terms of SABer, Experiment 2 examined whether it could generate useful semantic attributes that are made available to client applications, and also aimed to get some early feedback from domain experts on its interface and the functionality it offered. This experiment involved three separate client applications, each in a different domain (evaluating feature E4 described in Section 5.1), which were designed specifically for use with SARA. Each of these applications used SME that was encoded in a prototype of SABer. The final part of Experiment 2 was to gather reactions from the application developers as to their experiences of using SARA as a mediator between their software and the raw data sources. The collection of this feedback about SABer and SARA meant that further improvements to their design could be implemented in time for the next experiments. Furthermore, it helped highlight how SARA provided client applications with access to data sources without prescribing a specific user interaction paradigm for the GUI, or for their developers to know a specific query language (feature E1 from Section 5.1).

SARA is a middleware system and therefore it cannot be evaluated purely on its own. Hence, a third experiment involving two of the client applications used in Experiment 2 was devised. Experiment 3 centred on whether the improved version of SARA facilitated client applications to support users who are exploring information in domains of interest to them. In each experiment participants use the client applications, with each having a separate evaluation study afterwards. The feedback from users was used to determine whether they had found them useful. Furthermore, the impact of these client application experiments on the evaluation of SARA is also discussed, specifically in reference to features E1, E3 and E4 described in Section 5.1. If many of the aspects of the applications that users found useful were directly facilitated by SARA, it would provide further support that different client applications can benefit from using SARA. Furthermore, the features offered by SARA stem directly from the design requirements of the framework associated with the *expert-supported approach to data exploration*. Hence by showing that these features were useful and well received by end-users, it provided validation for the underlying approach and framework as well.

A theme of this thesis is that non-technical domain experts should be able to generate SME. Hence a fourth experiment was undertaken to examine if such experts could successfully generate semantic attributes in SABer i.e. that feature E2 described in Section 5.1 was being supported by SARA. The version of SABer used in this experiment had been updated and improved according to domain expert feedback in Experiment 2. This user trial examined the usability of SABer, and thus involved measuring efficiency (speed of creating semantic attributes), effectiveness (accuracy of semantic attributes created) and satisfaction on the System Usability Scale (see Appendix G) (Brooke 1996). Furthermore, this trial helped explore whether SABer (and by extension the Semantic Attribute Model) was sufficiently abstract so that users were not concerned with the differences between the various semantic attributes and their underlying data types. The functionality that SABer offers domain experts is rooted in the design requirements of the framework associated with the *expert-supported approach to data exploration*. By demonstrating that this functionality was successfully implemented and appreciated by end-users it provided further validation for the underlying approach and framework.

As SARA had iteratively improved as a result of the previous experiments, it was necessary to take the most recent version and provide a guideline to potential application developers as to the overhead they are likely to incur by using SARA. Hence Experiment 5 tests the performance of SARA (feature P1), in order to examine how quickly it processed Semantic Attribute Queries of varying complexity. Table 5-1 summarises the details of all the experiments presented in this thesis.

Table 5-1. Details of Thesis Experiments

	Name	**Evaluated Features**	**Evaluation Subjects**	**Technologies Employed**	**Evaluation Method**
Exp. 1	Multi-Source Music Trial	E1 & E3	End Users & Application Developers	SARA v1	User Trial and Questionnaire (Qualitative & Quantitative)
Exp. 2	Multi-Domain Evaluation of SARA and SABer	E1 & E4	Application Developers & Domain Experts	SABer v1 & SARA v1	Semi-Structured Interview with system users (Qualitative)
Exp. 3	End User Experiences with Client Applications	E1, E3 & E4	End Users	SARA v2	User Trial and Questionnaire (Qualitative)
Exp. 4	SABer User Trial	E2	Domain Experts	SABer v2	User Trial and Questionnaire (Quantitative) & SUS Test
Exp. 5	SARA Performance Evaluation	P1	N/A	SARA v2	Performance Evaluation (Quantitative)
Case Study	Music Domain Case Study	C1	N/A	SABer v2 & SARA v2	Case Study (Technical Analysis)

Furthermore, Figure 5-2 depicts the relationship between the third research objective (evaluation) and the individual evaluation experiments that they are related to (please note that the size of the individual boxes is not an indicator of their relative importance).

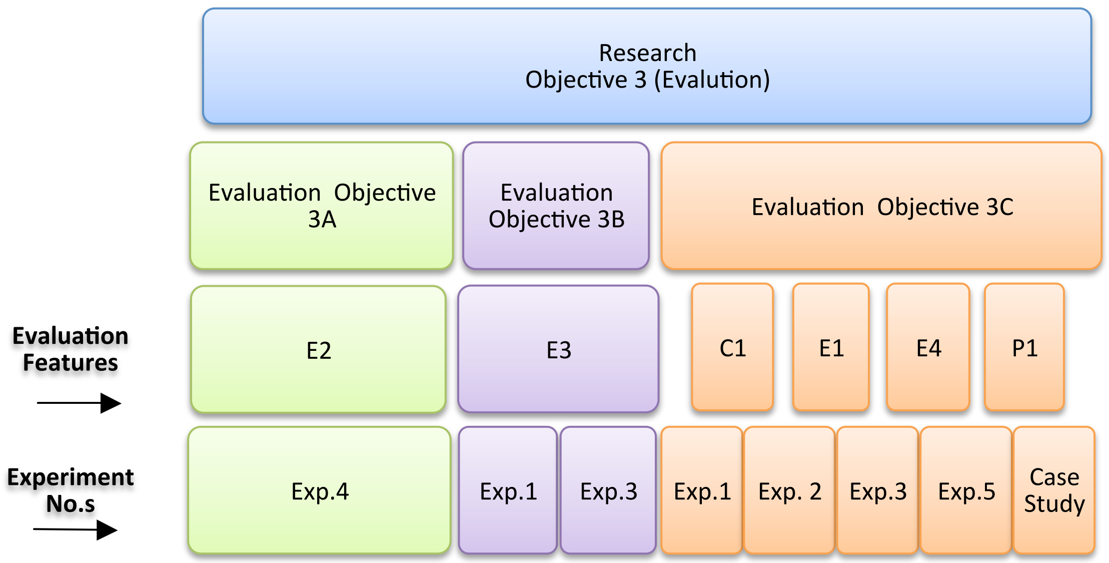


Figure 5-2. Relationship between Research Objective 3 and the Evaluation Experiments

5.3 Experiment 1: Multi-Source Music Trial

5.3.1 Experimental Goals

There were two main goals for this initial experiment. The first looked at whether the approach implemented in the SARA prototype, where it acted as a mediator between a client application and multiple data sources, was appropriate from a technical perspective. The second aim was to get some early qualitative feedback as to whether SARA facilitated useful functionality that users of a client application could exploit. Specifically this functionality was enabling application users to create complex queries over multiple separate data sources in order to explore the music domain. By showing this, it would help support the hypothesis that SARA provided client applications access to data sources without prescribing a specific user interaction paradigm for the GUI, or for their developers to know a specific query language (E1). Furthermore is would support the notion that SARA enabled casual users to appropriate (or tailor) SME within a client application and send complex queries to data sources (E3).

5.3.2 Experimental Setup

This experiment involved the use of an early prototype of SARA that worked exclusively with XML sources, as well as a purpose built client application that contained a query building interface. The client application communicated to SARA by passing parameters in accordance with SARA's API, and there were no restrictions placed on the design of the client's GUI. The implemented GUI enabled users to create consolidated queries across three music data sources in order to better explore the domain. However, the design of the client application itself was not evaluated in this experiment, as its role was purely to make SARA's functionality available to participants. The data sources used in this particular experiment were all stored in eXist XML databases and consisted of:

- Three separate iTunes databases totalling 7,000 songs.
- A UK singles and albums music chart archive from 1994-2008
- Data harvested from the web services offered by Last.fm

Before using the iTunes database some pre-processing was necessary in order to give each element a unique name. For example the name and artist of a song are represented in iTunes as key value pairs as follows:

```
<key>Name</key><string>One</string>
<key>Artist</key><string>U2</string>
```

An XSL transformation was thus used to change this structure to the following:

```
<SongTitle>One</SongTitle>
<ArtistName>U2</ArtistName>
```

The data harvested from Last.fm[39] (a popular online radio website) was gathered by collecting the 7,000 songs from the iTunes databases and for each one querying the Last.fm REST API to get five related songs, five related artists, whether the artist had concerts scheduled and where a song ranked in number of plays in comparison to the artist's other songs. This information gathered was then collated into a separate XML file.

After examining the metadata available from the different sources, a domain expert then created eleven semantic attributes for the personal music domain and had them integrated into SARA and the client application by a computer developer. The client application listed the names of these semantic attributes, and by selecting checkboxes beside the names, it enabled users to create queries from this SME. Furthermore, if a user wanted to tailor a specific semantic attribute, they clicked on the "personalise" button beside the relevant semantic attribute, and in a pop up window they inputted new values for the rule

[39] http://www.last.fm/api

e.g. deciding that *long duration* was greater than 300 seconds or *high popularity* was chart position less than 4. The semantic attributes created were as follows:

- Duration_by_specific_artist
- Duration_by_all_iTunes_artists
- Popularity_by_UK_Charts_1994-2008
- Popularity_by_Last.fm
- Related_songs
- Related_artist
- Touring_Artist
- Song_Recentness_by_artist
- Song_Recentness_by_all_iTunes_artists
- Genre
- Song_File_Quality

As SABer had not been fully developed at the time of this experiment, the domain expert in question used their computer programming skills to create the semantic attributes in XML. Hence the ease in which non-technical domain experts could create semantic attributes was not being analysed in this experiment.

Twelve users (four casual computer users and eight people with strong IT skills) participated in the experiment which involved the forming of ten complex queries in the client application. Seven of these users were male, five were female, eight were in their 20's, three were in their 30's and one was in their 40's. Once the experiment was completed all users were given a post questionnaire to complete. The first part of the questionnaire asked the participants whether SARA offered any benefits to end users interested in a particular domain. The second section was only compulsory for those participants with a computing background, and aimed at finding out ways of improving the functionality offered by SARA. This is because in the early stages of development of SARA, there was a focus on ensuring that the technologies employed and the API offered were both appropriate, and suitable for potential application developers. The entire process, from the explaining of the experiment through to the filling in of the questionnaire, took approximately forty minutes.

In the experiment, six tasks were given to the participants including *locating any songs in the iTunes collection that reached number 1 in the UK charts since 1994 and identifying five songs similar to these hits*. This particular task highlighted how users could create one consolidated query that previously would have involved the correlation of information from three separate data sources. Other tasks included finding *any artists in the collection that had concerts currently scheduled despite not having had a hit since 1994* and locating *any long songs in the collection by artists similar in style to the Beach Boys*. Users were also encouraged to tailor these queries to take into account their definition of a *long* song, or a user's preferred time span (1980-1990 etc.) or perception of a *hit* (top ten, top forty etc.). Users also devised their own queries which were executed by SARA.

5.3.3 Experimental Results

5.3.3.1 Satisfaction with SARA functionality

After all participants were finished interacting with the application they were given a questionnaire containing statements about SARA (see Appendix D). They were then asked to fill in, on a scale of 1-10, how much they agreed with each statement. The aim was to elicit information such as whether participants believed they had they had sufficient control tailoring the semantic attributes to their own interpretation, whether it was easy to compose semantic attributes into consolidated queries and if they gained knowledge from the system that would not have been easy to locate otherwise. Table 5-1 summarises the full findings of the questionnaire. The average agreement to all these statements was 8.9 out of 10, thus it could be concluded that participants were largely satisfied with the functionality offered by SARA and liked how it supported them in exploring a domain. Users were also asked if they could see the potential of such a system being used in a variety of domains. Many domains were mentioned by the users, including correlating gambling information, browsing media libraries, analyzing the performances of stocks over time and monitoring network errors.

The user questionnaire also indicated what users thought was successful about the current implementation of SARA. User comments consistently stated that SARA_supported them in creating complex queries in an easy fashion (e.g. "*the process was simple*", "*I liked that it was easy too [sic] send queries*"), and that correlations could be drawn between the results that would not have been easy to arrive at otherwise. For instance, it was noted that it was now viable to "*group together all the artists in the collection that had not had chart* *success recently*". Likewise it was said that "*comparisons could be drawn between online fans and the general charts, in how particular songs were rated*". Furthermore, users described how they felt that they had sufficient control in tailoring the subjective semantics in their searches so that the information returned was more aligned to their interpretation (e.g. "*I liked that I could change the durations to only return the really short songs*", "*I thought a hit should only be a number 1 song so changed those values*"). This feature facilitated them finding "*a needle in a haystack*". Even with only a small number of semantic attributes available in this experiment, users believed overwhelmingly that these provided a good initial starting point to explore the domain. However, if more sources and semantic attributes were made available to users, as well as a choice of domains, the benefit of a system like SARA would increase significantly. This is because the addition more sources and semantic attributes would allow a greater scope of queries to be formed, and thus increase the potential of interesting correlations being drawn across different information sources.

Table 5-2. Results of Multi-Source Music Trial Questionnaire

No.	Statement	Avg. Agreement Score out of 10	Standard Deviation
1.	Using this framework is a more efficient way of finding information than having to consult the individual data sources separately?	9.0	0.76
2.	Using this framework it is easy to combine data from different sources?	8.9	0.62
3.	Using this framework allows users to combine and interchange attributes (popularity/duration/freshness etc) easily?	8.6	0.77
4.	Using this framework enables knowledge to be gained that wouldn't be possible by consulting just a single source?	9.6	0.51
5.	This framework sufficiently enables users to personalise and tailor their queries (a very popular song is top 10; a long song is more than 5mins, etc.)	8.8	0.61
6.	The approach used by the framework is very applicable to other domains beside music?	8.6	0.70

5.3.3.2 Technical Analysis

Given the experimental results, it could be determined that SARA had been a technical success in terms of enabling participants to query (via a client application) multiple sources from a domain in a consolidated fashion. However, from the analysis of the questionnaire results, it was clear that SARA would have to support more features in order to become a really useful and flexible architecture. For instance, if domain expertise is to be captured and encoded as a semantic attribute, then it would be useful if this could be achieved via an authoring tool without the help of a computer developer. This would open up the creation of semantic attributes to non-technical domain experts and make the adoption of SARA more attractive. Moreover, these semantic attributes need to be represented as a model that is transferable between different installations.

Another limitation of the SARA prototype was its exclusive interaction with XML data stored in databases. In order to open up its functionality to a much wider range of data, it was stated that SARA should interact with data of different formats that are stored remotely as well as locally. Furthermore, by allowing data to reside remotely it would also mean that much time consuming pre-processing of data (e.g. the *a priori* querying of Last.fm to generate additional data on songs) would be greatly reduced.

5.3.4 Summary

Through this early user trial it was concluded that the functionality offered by the SARA prototype was useful to end users and allowed them to explore multiple sources from domain. Specifically SARA enabled users to appropriate and tailor semantic attributes in order to send complex queries to multiple data sources (feature E3). Furthermore, its applicability to a variety of domains was highlighted, and the benefits of expanding the system to include more semantic attributes and data sources were noted. In terms of the technology itself, SARA enabled a client application to communicate with the raw data sources successfully, and showed how the API did not prescribe a specific GUI for the client application, nor a specific query language to be used (feature E1). However a number of suggestions on how to improve SARA were noted (e.g. support multiple data types, use remote sources and develop an SME authoring tool), and these suggestions were incorporated into the ongoing development of *the expert-supported approach to data exploration*, as well as the later implementations built on top of it

5.4 Experiment 2: Multi-domain Evaluation of SABer and SARA

5.4.1 Goals of Experiment

One of the design compromises of SARA is that end-users are limited, when browsing a domain, to combining and tailoring the semantic attributes created by experts. Hence, it is vital that the manual step of creating semantic attributes is easy to do and not overly time consuming, so that a wide selection of high quality semantic attributes are made available to end users. One of the key outcomes from the first experiment was the reaffirmation of the need for an authoring tool to help these experts create semantic attributes and deploy them within SARA. Hence, a prototype of such an authoring tool called SABer (Semantic Attribute Builder) was built to provide this functionality.

The first of three goals in this experiment was to show that useful semantic attributes could be generated by SABer and then be deployed within SARA installations of different domains (highlighting feature E4, which states that SARA and SABer should be domain independent). The second aim of this experiment was to get some early feedback on SABer's interface and the functionality it offered. The final goal was to get reactions from application developers as to their experiences of using SARA as a mediator between their software and the raw data sources. By collecting this feedback on both SABer and SARA, further improvements to their designs could be implemented in time for the later experiments. Furthermore, it would help highlight how SARA provided client applications with access to data sources without prescribing a specific user interaction paradigm for the GUI, or that their developers needed to know specific underlying query languages (feature E1 described in Section 5.1).

5.4.2 Experimental Setup:

The first three participants to use a prototype of SABer all had computer coding experience, as well as some expertise within their domain of interest. Their expertise was in the digital images, film domains and academic publications respectively. In order to place any semantic attributes they created in an authentic environment, these users all developed a separate client application for each domain. These applications used functionality offered by SARA to connect with the underlying information sources relevant to their domain. The three applications developed are now described in turn.

In the academic publications domain, a highly visual prototype application was developed in Adobe Flex that uses semantic attributes and the SARA infrastructure to support a more explorative approach to finding relevant publications. It contains a test set of 300 publications from the Knowledge and Data Engineering Group's website [40] and their

[40] http://kdeg.cs.tcd.ie/publications

associated metadata. The semantic attributes created provide a number of search axes under which users could search and which are complimented by the novel visualisation interface developed for the client application. Figure 5-3 shows the GUI of this client application.

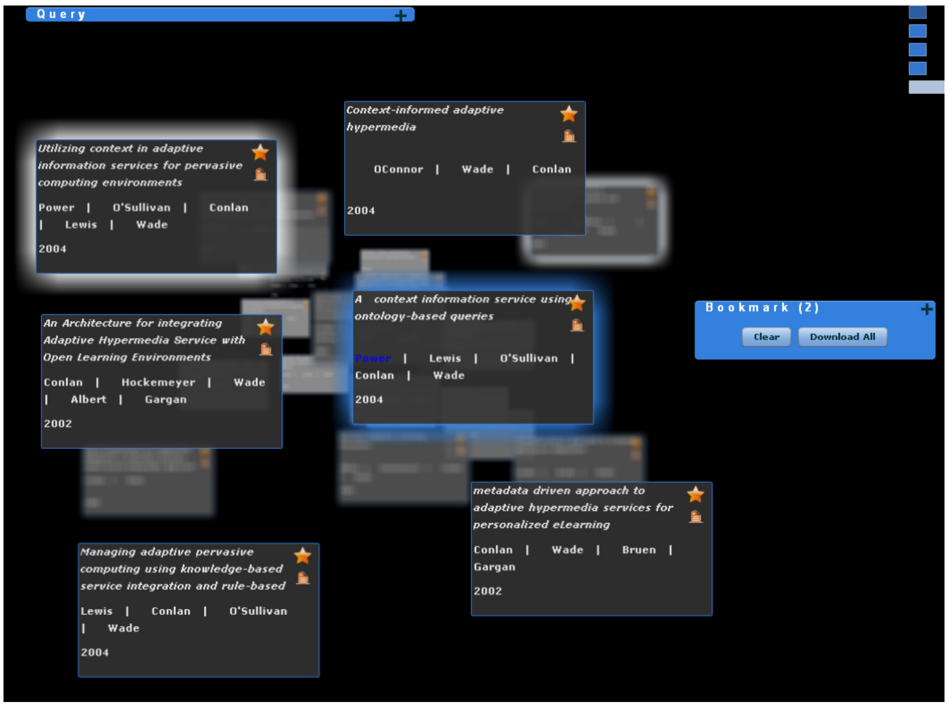

Query
Utilizing context in adaptive information services for pervasive computing environments
Power | O'Sullivan | Conlan | Lewis | Wade
2004
Context-informed adaptive hypermedia
OConnor | Wade | Conlan
2004
An Architecture for integrating Adaptive Hypermedia Service with Open Learning Environments
Conlan | Hockemeyer | Wade | Albert | Gargan
2002
A context information service using ontology-based queries
Power | Lewis | O'Sullivan | Conlan | Wade
2004
Bookmark (2)
Clear
Download All
metadata driven approach to adaptive hypermedia services for personalized eLearning
Conlan | Wade | Bruen | Gargan
2002
Managing adaptive pervasive computing using knowledge-based service integration and rule-based
Lewis | Conlan | O'Sullivan | Wade
2004

Figure 5-3. GUI of Client Application that Explored Academic Publication Domain

X2Photo (Gürel 2009) is an application developed in Adobe Flex[41] that helps users browse image repositories with reference to the aesthetics of the photographs, as well as their content. It uses a data source of over 12,000 photographs and has access to technical metadata of each image (hue, saturation, lightness values etc.) as well as its metadata and tags taken from the Flickr[42] website. The *expert* rules, based on Colour Theory (Parramon 1989), were encoded using SABer, and end users were able to leverage this knowledge while exploring the photograph repository for relevant images. X2Photo employs a novel browsing interface consisting mainly of a wall of draggable photos. Figure 5-4 shows the front-end of the system and depicts the three main areas of the interface. The main part of the screen is called the Discovery Space and contains the result set of photographs retrieved for a user query. The wall of photographs can be dragged by the user and individual images can be selected their associated tags and semantic attributes.

[41] http://www.adobe.com/products/flex/
[42] http://www.flickr.com

The bottom of the screen is dominated by the AttBar which represents each of the nine semantic attributes (from Table 5-3) as a vertical bar. The process involved in creating these semantic attributes is described in Section 5.4.3.1, and the success of X2Photo is largely dependent on how well these semantic attributes are designed. If the semantic attributes are of poor quality, the SME features offered by SARA are unlikely to be of great benefit to the users of X2Photo. Each vertical bar in the user interface contains the different parameters relating to each semantic attribute, with Figure 5.4 showing the results from a query containing the aesthetics; *Lively*, *Luminous* and *Cool* from the AttBar. The user can select one parameter from each bar to run a query, and can decide if the AttBar is visible or not by clicking on its icon button.

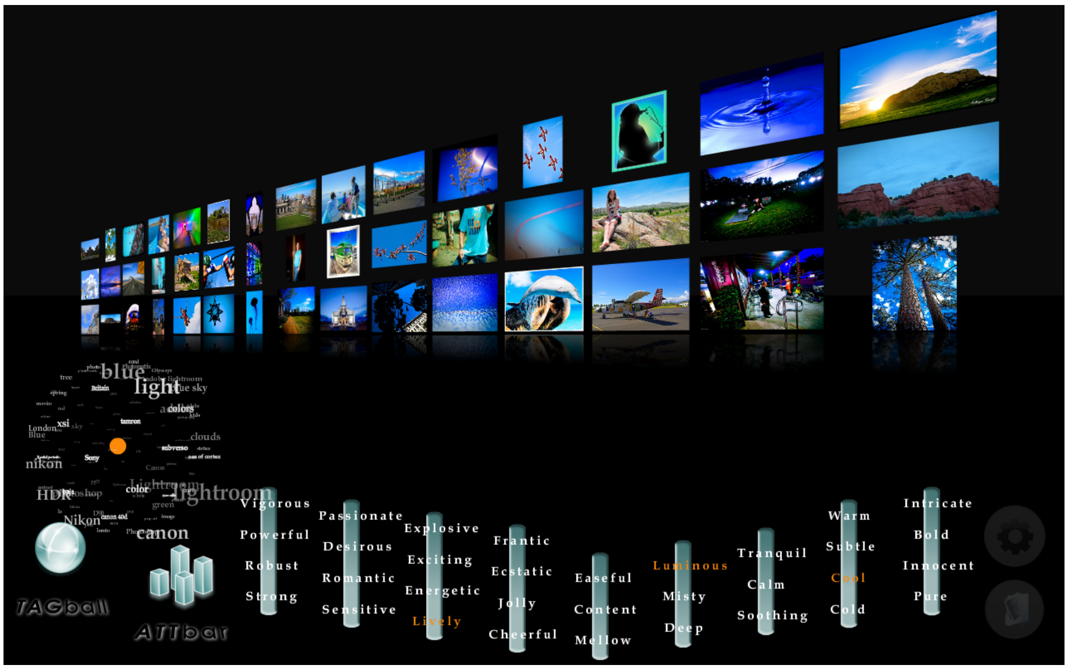


Figure 5-4. The X2Photo Interface

In order to help the user find an image that has specific content, the system shows any tags associated with the result collection of photographs, as well as those from each individual photograph. By integrating this with support for the aesthetic exploration of images, users are given a more flexible way of finding relevant photographs. The number of crowd sourced tags typically exceeds the number that can be clearly displayed with a simple tag cloud, hence a TagBall is used instead to allow large numbers of tags to be displayed without cluttering the UI. In the bottom left hand corner of Figure 5-4 is the TagBall which displays all the Flickr tags related to the entire result set or individual photograph. The user can simply use any selection of these to refine their searches.

The Film Domain Exploration Client (Hengster 2009) is an application developed in Adobe Flex that lets users explore the film domain, via the different relationships that exist between the films, and to build up collections of films. Semantic attributes, relating to different ways of measuring the *success* of a film, were created in SABer from different data sources (including the Internet Movie Database[43], Rotten Tomatoes[44] and Freebase[45]), and used to help implicitly model the preferences of users and to provide recommendations to similar films. The Film Domain Exploration Client employs a novel user interface, where users select film posters to trigger queries. Furthermore, in this application the semantic attributes are not displayed to users but rather used in the background as similarity metrics for the films. Figure 5-5 shows the GUI of this client application.

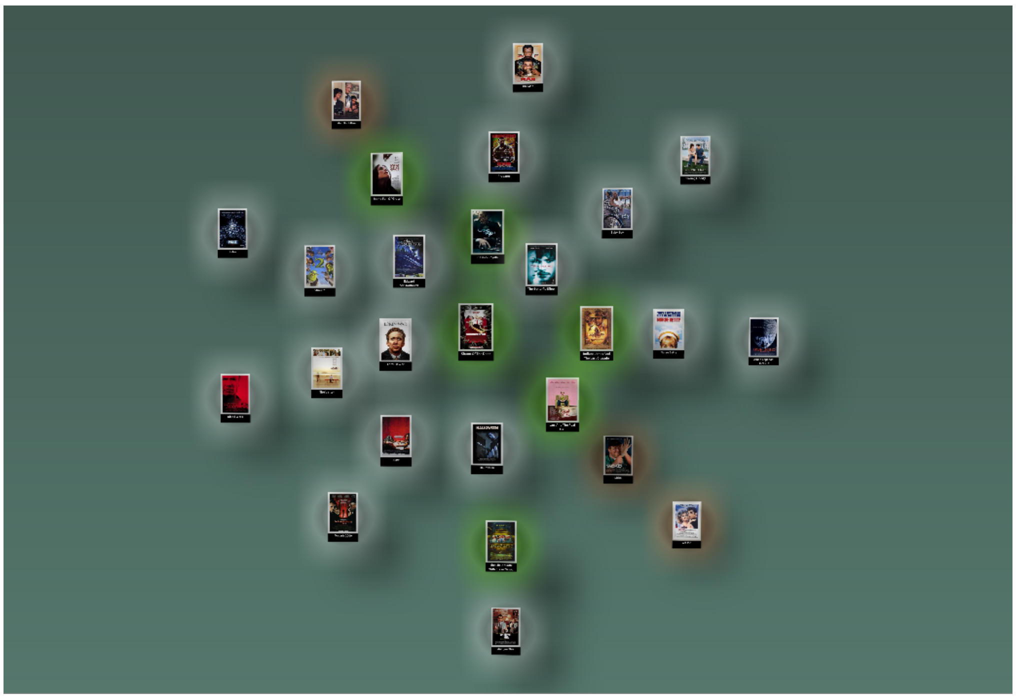


Figure 5-5. Screenshot of Film Domain Exploration Client

The first step in the procedure of using the application involves selecting an initial film poster by typing in a film name. This initial film is used by the application to calculate and display 20-30 related films in a spiral around the focus movie at the centre of the stage. This number of films means that the user is not overwhelmed by too many objects at once, and by having a number higher than 8 or 10 gives the application enough diversity to grab sufficient attention and to support the explorative spirit of the application (Moreno 2004). In order to ensure that a user is able to recognise his preference movie, the posters are

[43] http://www.imdb.com
[44] http://www.rottentomatoes.com
[45] http://www.freebase.com

placed onscreen one by one starting in the centre and continuing along a clockwise spiral path. The user is encouraged to explore the films by zooming and panning as required and rating the films presented.

The application developer felt that rating a film poster was central to the success of the system, as it helps the reasoning within the personalisation and modelling components. Hence, the rating metaphor had to be easy to comprehend and also be consistent. There were three different degrees a rating could have: like, dislike and neutral. Every poster starts in the neutral phase and when a user pushes the poster away it receives a red outline and indicates they dislike the film. When they pull a poster closer it gets a green outline and indicates that they like the film. Should a poster that has been rated previously within the current session come up again, the rating is restored and the poster will be shown with the according visual adjustments. This makes it easy to recognise any previous actions. Moreover, three collections of posters (one for each of liked, neutral and disliked films) are always accessible by zooming out, so it is possible for the user to keep track of what he has collected during his exploration. Finally, to encourage continuous exploration, a user can always refocus the interface on any of the films displayed on the stage. This centres that movie poster and orbits related movies around it according to the various semantic attributes created for the application. Thus films with similar grossing, IMDB rating, profit or awards are rendered onscreen around the focus movie.

The semantic attributes generated in each of the three applications just introduced are described in detail in the next section. Once the applications were finished a semi-structured interview took place with the three participants to gather feedback on the process of generating semantic attributes with SABer (see Appendix E). Furthermore, their experiences of using the SARA API to mediate between their applications and the data sources were also recorded. The interviews took approximately twenty minutes to perform in each case.

5.4.3 Experimental Results

This section describes the semantic attributes generated by the three participants for their respective applications, and then summarises the participants' experiences of working with SARA and SABer as documented in structured interviews.

5.4.3.1 Digital Imaging Domain

The expert vocabulary created in SABer for describing images in X2Photo consisted of ten semantic attributes, with various numbers of parameters. *Temperature* was one such semantic attribute, and this was quantised by four different parameters (*Warm, Subtle, Cool and Cold)* according to Colour Theory (Parramon 1989). Colour Theory relates colours to temperatures, for example it specifies that *blue* and *green* are *cool* colours, while *red* and *orange* are *warm*. Furthermore, because the hue of a colour can be represented as an angle of the colour circle, when the colour circle is divided into twelve equal intervals; on each 30° angle you will find the following colours (key colours in bold):

1. **red**
2. red-yellow (orange)
3. **yellow**
4. yellow-green
5. **green**
6. green-cyan
7. **cyan**
8. cyan-blue
9. **blue**
10. blue-magenta
11. **magenta**
12. magenta-red

Red, *yellow*, *green*, *cyan*, *blue*, and *magenta* are regarded as the key colours, each having intermediate colours between them. Each of the photographs available to X2Photo contained metadata relating to the Hue, Saturation and Light values, which allowed *expert* rules to be generated for them in SABer. For instance, when constructing the *Temperature* semantic attribute, the expert could quantise it into four parameters as follows, where H, S and L represent Hue, Saturation and Lightness respectively:

$$\textbf{\textit{WARM}} = \{ (0 \le \text{H} < 75 \text{ AND } 15 \le \text{L} \le 90) \text{ OR } (\text{H} \ge 300 \text{ AND } 65 \le \text{L} \le 90) \} \text{AND } (\text{S} \ge 25)$$

$$\textbf{\textit{SUBTLE}} = (75 \le \text{H} < 120) \text{ AND } (15 \le \text{L} \le 90) \text{ AND } (\text{S} \ge 25)$$

$$\textbf{\textit{COOL}} = (120 \le \text{H} < 210) \text{ AND } (15 \le \text{L} \le 90) \text{ AND } (\text{S} \ge 25)$$

$$\textbf{\textit{COLD}} = (210 \le \text{H} < 300) \text{ AND } (15 \le \text{L} \le 90) \text{ AND } (\text{S} \ge 25)$$

Using these rules, a photograph is assigned a temperature description (warm, cool etc.) based on which equation its HSL colour space satisfies. Table 5-3 lists all ten semantic attributes and their associated parameters created for X2Photo in SABer. There were nine heavily subjective semantic attributes created, as well as one objective semantic attribute named *has tag called*, which allowed users to specify tags from Flickr that the end images should have associated with them.

Table 5-3. The Ten Semantic Attributes and their Parameters Created for X2Photo

1. Power	**2. Passion**	**3. Energy**	**4. Joy**	**5. Ease**
Vigorous Powerful Robust Strong	Passionate Desirous Romantic Sensitive	Explosive Exciting Energetic Lively	Frantic Ecstatic Jolly Cheerful	Easeful Content Mellow
6. Light	**7. Blue**	**8. Temperature**	**9. Purity**	**10. has tag called**
Luminous Misty Deep	Tranquil Calm Soothing	Warm Subtle Cool Cold	Intricate Bold Innocent Pure	Default

All the semantic attributes in Table 5-3 were created in SABer in a similar fashion to *Temperature* (*has tag called* being the exception) and then integrated into X2Photo. This meant that the semantic attributes could be joined together into a complex search by end users e.g. *Return all images that are Calm, Cool and Misty and that have a tag called 'Boat' or 'Fisherman'*.

X2Photo received positive feedback overall when it was evaluated (see Appendix H for more details). Moreover users found that the more natural expressiveness of the semantic attributes gave them much more freedom when searching for images, compared to when they were limited to solely relying on search terms in traditional keyword searching. Thus injecting expert knowledge, based on the aggregation of raw low-level data, into a conventional system which only supported tag-based search, proved successful. It allowed users to go beyond simply expressing the content of an image, and express the actual aesthetics of the image more easily. This was largely facilitated through the use of SARA and the experiment shows how SME, captured in SABer, can be leveraged by ordinary users within a client application.

5.4.3.2 Film Domain

The semantic attributes created in SABer, by the domain expert, for the Film Domain Exploration Client were all ways of measuring the *success* of a film and were as follows:

- IMDb_Film_Rating
- Worldwide_Box_Office_Grossing
- Award_Winner_in_Oscars_BAFTAs_Cannes_Berlin_or_MTV
- Grossing_to_Budget_Ratio

During the development phase a number of additional semantic attributes were also created in SABer and integrated into the application. These semantic attributes allowed querying for related films based on several extra axes such as by:

- cinematographer
- director
- producer
- actor
- film_genre
- year_released.

However, these semantic attributes were disabled in the evaluated version of the application in order to concentrate on the film *success* semantic attributes listed above. In contrast to X2Photo, none of the semantic attributes were actually displayed in any way to users in the Film Domain Exploration Client. Instead the semantic attributes were used in the background as similarity metrics with which to find related films. The user experience with the application was positive overall (see Appendix I for details), and it showed how domain expertise encoded in SABer could be used to support subtle domain exploration as well as users who explicitly wished to create complex queries.

5.4.3.3 Academic Publications Domain

The semantic attributes created in SABer for the academic publications application covered the following characteristics:

- Author
- Year
- Title
- Type of Publication
- Theme

Despite the limited range of metadata with which this application had to work with, these basic semantic attributes facilitated interesting explorations of the domain nonetheless. They helped complex queries to be formed by end users and certain relationships between papers to become apparent e.g. how the themes of papers published in the group had evolved in the previous ten years. This showed how visualisation applications can benefit from using SARA to connect with the underlying data.

5.4.3.4 Feedback on SABer and SARA

As mentioned previously, the three domain experts all had computer coding experience and worked with an early prototype of SABer (version 1) to generate their respective semantic attributes. The fact that all three participants successfully used SABer to produce semantic attributes for their client applications shows that SABer had successfully supported them in encoding SME. The information gathered from their individual interviews was later used to improve future releases of SABer that specifically aimed to support non-technical experts as well those with computing skills.

In general feedback was very positive and SABer was found to be easy to use, though a number of changes and new features were suggested. The most important suggestions were prioritised and implemented in the next version of SABer:

- The GUI of SABer was refined in specific areas as per comments of the domain experts, and the tool now provides more of a step-through feel for users when creating semantic attributes.
- The need to define variables in SABer as part of the *template* rules was removed. This will particularly be of benefit to non-technical users who would not be immediately familiar with the concept of a variable.
- More operators and functions were added to the rule generation section in SABer to give users more flexibility.

The following changes were noted for implementation in future releases of SABer:

- Allow free text rule input in SABer for those advanced users with significant coding experience in the underlying query language.
- Support visualisation of semantic attributes already created in SABer so that they can be organised and edited through the GUI.
- Improve how parentheses are visualised within the rule generation section of SABer.

In terms of SARA's API, the following suggestions by developers were implemented in the next release of SARA:

- The API call to send queries to SARA was simplified as per request of application developers who thought the original was slightly cumbersome.
- A set union operator for Semantic Attribute Queries was added, as applications were previously limited to joining semantic attributes via set intersection or set difference.

In the course of the interviews one of the client developers noted that it was *"really nice to have SARA as it meant I didn't have to worry about the underlying data and its structures that can be very messy to work with… I just had to access a single framework that made all* *the relevant information available to me".* Another one thought that *"creating the semantic attributes and then using them to form compound queries was easy".* Interestingly one of the domain experts noted that the "*rule creation in the tool was quite similar to the process involved in generating smart playlists within iTunes".* Since as successful commercial application as iTunes had a superficial similarity for non-technical users to generate rules, he believed *"that it would be quite easy for a general audience to work with the SARA authoring tool".* Finally, all three developers thought that the API in its current form was easy to use, though some improvements could be made to make it more streamlined. This showed that a parameter based API could successfully allow applications to interact with SARA, and that the developers did not have to learn the underlying query language associated with each data source (feature E1 described in Section 5.1).

5.4.4 Summary

This section has described how SARA's features supported three separate applications working in different domains in accessing data sources for the benefit of their users. Moreover, it highlighted how SABer can be used by domain experts to capture domain expertise, which is then deployed in SARA for end users to leverage using client applications. It was also shown how domain experts can use SABer to create useful semantic attributes deployed within applications, thus supporting the contention that SABer is a generic tool that helps expertise in metadata rich domains to be encoded. Therefore feature E4 (SARA and SABer should be domain independent) was successfully implemented.

As each of the three applications employed completely different styles of GUI (the academic publications application combined a query builder with a 3D rendering of results; the film domain application did not allow users to explicitly build queries, but rather rendered similar films in a spiral according to their similarity with the target film; and the digital imaging application combined TagBall, AttBar and Discovery Space Elements into a novel interface), this experiment also demonstrated how SARA does not prescribe applications with a specific interaction paradigm (feature E1). Furthermore, by gathering feedback from the users of SABer and of those using SARA's API, it meant that the most important suggestions were implemented in the next releases of both systems. The following section will now describe in detail the end user feedback from experiments performed with X2Photo and the Film Domain Exploration Client. The aim of this is to analyse whether the features offered by SARA enabled the client applications to provide functionality to end users that was well received.

5.5 Experiment 3: End User Experiences with Client Applications

5.5.1 Introduction

The main motivation behind SARA was to support users in exploring domain information in a more meaningful way. However, because SARA is a middleware system, it cannot be evaluated purely on its own. Hence, X2Photo and the Film Domain Exploration Client, described in the previous experiment, were selected for further user trials. The difficulties in doing quantitative evaluation of exploratory search systems, in comparison to the standard precision and recall metrics available in IR systems are well documented (White et al. 2006; Qu & Furnas 2008). Hence, these user trials were focused more on getting qualitative feedback about the applications, and (of most importance to this thesis) analysing whether SARA facilitated any of the features that were well received by users. The individual aims of each user trial will be described in their respective sections, but the overall functionality under examination in terms of SARA was that:

- it provided client applications with access to data sources without prescribing a specific user interaction paradigm for the GUI (E1)
- casual users could use semantic attributes within a client application to send useful queries to data sources via SARA (E3)
- it was domain independent (E4)

This functionality offered by SARA stems directly from the design requirements of the framework associated with the *expert-supported approach to data exploration*. Hence by showing that these features were useful and well received by end-users, it provided validation for the underlying approach as well. This section is divided into two parts; the X2Photo user trial is described first, followed by a description of the Film Domain Exploration Client user trial. A discussion on how these two user trials helped evaluate SARA then follows.

5.5.2 Goals of X2Photo User Trial

The aim of this experiment was to compare users' experiences of finding similar photographs using the X2Photo application with their experience of using the Flickr[46] website. X2Photo uses SARA and the semantic attributes created in SABer to help users search for photographs by their aesthetics as well as by their content. Hence its comparison with Flickr's traditional keyword matching search box provided an interesting contrast. The description of this user evaluation itself has been summarised from the M.Sc. thesis (Gürel 2009) that it was a constituent part of, and the trial was conducted by the M.Sc. candidate themselves. Details of how the experiment was conducted can be found in Appendix H. In terms of SARA, this experiment was used to test if its features provided client applications with useful functionality which was well received by end users. This would help support the notion that SARA is a good middleware system. Specifically this experiment looked at how SARA facilitates the development of novel user interfaces (feature E1) and allows users to explore information domains by generating queries using semantic attributes (feature E3).

5.5.2.1 Analysis

The experiment with X2Photo showed that the semantic attributes available in the system gave users additional useful axes when searching for photographs. This indicates that injecting expert knowledge, based on the manipulation of raw low-level data, into a conventional system which only supports tag-based search, can help users to more freely express both the photograph and the picture it is conveying. Even though a specific expert vocabulary may not be suitable or correct for each individual, users can adapt to the expert's view or choose to subscribe altogether to a different expert. A further option would be to use SARA's tailoring feature, so that end users could adapt the rules behind an expert's semantic attribute to better fit their own vocabulary or perceptions.

Users suggested that they should be able to subscribe to different experts and that there should be a more comprehensive range of semantic attributes. This all indicates that the users understood the aim of the tool and how it could be extended further. All users agreed that they could see a real-life application of the tool if some further improvements were

[46] http://www.flickr.com/

made and they could see themselves utilizing such a tool in their everyday lives. In order to offer users an alternative way of finding a photograph, a system has to have a rich vocabulary. Hence, increasing the range of low level features would enable experts to create more refined semantic attributes, resulting in a more useful system for end users. The successful feedback on X2Photo highlights how SARA and its semantic attributes can support applications that help users to explore and find relevant information, even in quite subjective areas such as photograph aesthetics. Specifically this shows that feature E3 had been implemented well, because users could use semantic attributes within a client application to send useful queries to data sources via SARA. Finally SARA did not prescribe any specific interaction paradigm for X2Photo other than requiring the parameters sent to it conformed to its API specification (feature E1 from Section 5.1). Hence, on top of the semantic richness that SARA provided X2Photo with, the freedom of design that SARA offered X2Photo facilitated a novel interface to be generated.

5.5.3 Goals for Film Domain Exploration Client User Trial

The main aim of this experiment was to see if semantic attributes could be used to implicitly model user preferences, and to see if the Film Domain Exploration Client provided users with an enjoyable and useful browsing experience of the film domain. The main functionality of the application was to use SARA and the semantic attributes created in SABer in order to present users with similar films to the film they had targeted. These recommendations were based on a similarity metric derived from each of the films four *success* axes. Each of these *success* axes were encoded as semantic attributes in SABer and stored in SARA. The description of this user evaluation itself has been summarised from the M.Sc. thesis (Hengster 2009) that it was a constituent part of, and the trial was conducted by the M.Sc. candidate themselves. Details of how the experiment was conducted can be found in Appendix I. In terms of SARA, this experiment was used to test if its features gave client applications useful functionality that was well received by end users. This would help support the notion that SARA is a good middleware system. Specifically this experiment looked at how SARA facilitates novel user interfaces to be developed (feature E1) and allows users to explore information domains by implicitly generating queries with semantic attributes (feature E3)

5.5.3.1 Analysis

SARA did not prescribe any specific interaction paradigm for the Film Domain Exploration Client apart from the parameters sent to it having to conform to its API specification (feature E1 from Section 5.1). Hence, on top of the similarity metrics that SARA provided the application with, the freedom of design that SARA offered Film Domain Exploration Client facilitated a well received interface to be developed. The group members also agreed that they were able to complete the task of finding films they liked and felt a strong immersion while doing so. This shows that semantic attributes can be deployed to help implicitly model user preferences, and used to support users in an enjoyable exploration of a specific domain. Specifically this shows that feature E3 had been implemented well, because users could use semantic attributes within a client application to send useful queries to data sources via SARA.

5.5.4 Discussion on how User Trials Helped to Evaluate SARA

As already mentioned, the two user trials just described were not only an examination of X2Photo and the Film Domain Exploration Client, but also of SARA, which was a key technology employed by both systems. It was vital to see if SARA facilitated features in the two applications that were well received by users. Specifically the functionality under examination in terms of SARA was that:

- it provided client applications access to data sources without prescribing a specific user interaction paradigm for the GUI (E1)
- casual users could use semantic attributes within a client application to send useful queries to data sources via SARA (E3)
- it was domain independent (E4)

The first noteworthy conclusion from these experiments is that both applications worked in completely separate domains with different metadata; photographs and films. Sections 5.3 and 5.4.2 already discussed SARA's application in the music and academic publication domains respectively, so it can be concluded that SARA is indeed a domain independent framework suitable for domains with rich metadata. This is one of the key features under examination (E4). Some of the user comments in the experiments mentioned that a wider selection of semantic attributes (built from extra metadata or low level features) would help improve the individual applications. One of the features of SARA's design is that it is extensible in terms of the semantic attributes and data sources it employs. Hence, it can support the addition of new semantic attributes, and can make these available to client applications for incorporation into their system. Moreover, because the models that SARA employs are reusable, semantic attributes or data sources previously defined elsewhere can be reused and quickly incorporated by application developers. This makes SARA an attractive system for developers who wish to evolve their applications over time.

One thing that was immediately clear from both user trials was that users overwhelmingly enjoyed the novel visualisation interfaces that they used to explore the respective domains. The creation of individual styles for these engaging interfaces was helped by the fact that SARA only required that the parameters sent to it from the client applications conformed to its API. This was one of the key features under examination (E1) and meant that developers got the benefit of supporting their users via semantic attributes, without major restrictions hindering their GUI design. In contrast, other approaches to supporting casual users to engage with multiple sources from a domain can provide severe restrictions on the interface displayed to users e.g. requiring a faceted browsing interface or a mashup of predefined widgets.

As well as finding the novel interfaces attractive, both user trials endorsed the notion that users had largely found the domain exploration that was facilitated by the client applications as useful. The explicit selection of semantic attributes in X2Photo to form a query, as well as the indirect choosing of semantic attributes in the Film Domain Exploration Client (by focusing on a specific film) were both successful mechanisms for supporting users to engage with the domains in question. This highlighted a key feature supported by SARA, namely that casual users can use semantic attributes within a client application to send useful queries to data sources (E3). SARA also has functionality that could quickly expand the features offered by both applications. For instance, SARA could enable X2Photo to use its tailoring feature, so that end users are able to adapt the rules behind an expert's semantic attribute to better fit their own vocabulary or perceptions. This would provide more options to experienced users of the domain. Furthermore, this approach could be used in a variant of the Film Domain Exploration Client, where users get to explicitly choose and tailor the film-based semantic attributes to form specific queries. Overall, based on the findings of these two user trials, it could be concluded that SARA was an effective and versatile middleware system. Furthermore, the functionality offered by SARA stems directly from the design requirements of the framework associated with the *expert-supported approach to data exploration*. This close coupling of design and implementation meant that SARA's successful evaluation also helped to validate the underlying approach and framework.

5.5.5 Summary

This section described two applications in different domains that were connected to SARA, as well as their evaluations by end users. These evaluations showed that participants enjoyed using these applications to construct complex queries (explicitly and implicitly) over different information sources and to explore the data in a useful fashion. Because this functionality was largely facilitated by SARA, it supports the contention that SARA is a useful framework for supporting the querying and exploration of metadata rich domains.

Though this section has highlighted how SME encoded in SABer can be deployed in SARA to provide useful functionality for client applications, it was not yet clear if experts without computer coding experience could use SABer effectively. This is of vital importance if SARA is to be applicable to a wide range of information domains, because the easy generation of useful semantic attributes means applications will have a wider range of features with which their users can explore a domain. The following section describes a discrete user trial with SABer that compares how non-technical and technical users perform when working with it. The main aim of this is to see if non-technical users can use SABer effectively.

5.6 Experiment 4: SABer User Trial

5.6.1 Experimental Goals

For SARA to be applicable to many domains it needs to be shown that non-technical domain experts can successfully generate semantic attributes in SABer. As stated previously, for the purposes of this thesis 'non-technical' people refers to computer literate participants with basic skills such as operating Internet browsers, but with no computer programming experience. Likewise, this experiment considered those participants with a degree in computer science to be 'technical' users. Section 5.4 described how technical domain experts can use SABer to create useful semantic attributes that are subsequently deployed in applications. The goals of the experiment described in this section are to measure the usability of SABer and its effectiveness in supporting non-technical domain experts to generate semantic attributes (feature E2 from Section 5.1). Furthermore, this experiment helps explore whether SABer (and by extension the Semantic Attribute Model) is sufficiently abstract to distance the user from the differences in the various semantic attributes and their underlying data types. The functionality that SABer offers domain experts is rooted in the design requirements of the framework associated with the *expert-supported approach to data exploration*. By demonstrating that this functionality is successfully implemented and appreciated by end-users, it will also help validate the underlying approach itself.

5.6.2 Experimental Setup

Two groups of twelve participants each were assembled (one group technical and the other non-technical), with each person engaging in the experiment separately and in isolation from other participants. In the technical groups, nine of the participants were male and three were female, with five participants in their 20's, six in their 30's and one in their 40's. In the non-technical groups, six of the participants were male and six were female, with six participants in their 20's, three in their 30's and three in their 40's. An entire session including the demonstration, performance of tasks and filling in of questionnaires typically took around forty minutes to complete. The first step of the experiment consisted of a demonstration of how to create 3 different semantic attributes in the music domain. This demonstration was done by the evaluator who inputted the semantic attribute details from a task sheet into SABer. These three tasks involved an identical process to that which the participants would undertake in the experiment. As described in Section 3.4.1 SABer supports semantic attributes of three different types (*expert*, *template* and *hybrid*) and of three data types (*XML, RDF and Web Services with a native API*). Each of the three tasks demonstrated to the participants involved a combination of a different semantic attribute type with one of the different data types.

Once these three tasks were demonstrated, the participants were given the same set of nine semantic attributes (each of varying complexity) to create in SABer. These semantic attributes were presented to the participants in a random order. This is because users tend to get quicker with later tasks when they are more familiar with the application interface. By presenting the tasks in a random order it meant that the average time taken to create a semantic attribute would not be lengthened due to it appearing near the start of the list, or shortened due to being near the end of the list. The semantic attributes presented to the user are listed as follows:

1. Quality_Of_Audio_Files_in_my_iTunes_Collection
2. Top_Singles_in_US_in_1990s
3. Artists_of_a_Genre_who_had_US_single_ that_Reached_a_Specific_Position
4. Countries_Paul_McCartney_has_Concerts_Scheduled_in
5. Artists_Scheduled_to_Play_Iceland
6. Popular_Beatles_Songs_According_to_Last.fm
7. Songs_On_A_Specific_Album_By_Specific_Artist
8. Countries_with_Very_Popular_Artists_on_MySpace
9. Irish_Artists_Popularity_on_MySpace

These semantic attributes spanned the following five sources:

1. An iTunes music library with over 30,000 songs stored in XML in an eXist database
2. The US singles charts from 1950-2008 stored as XML in an eXist database
3. The freebase.com music SPARQL endpoint[47]
4. The MySpace.com SPARQL endpoint[48]
5. Last.fm web services[49]

Figure 5-8 shows an example of the Irish_Artists_Popularity_on_MySpace semantic attribute that participants had to input from the task sheet into SABer during the experiment. Element data types were labelled as *Data Type A, B or C* rather than XML, Web Service API and RDF respectively, so that the technical participants wouldn't have pre-conceived perceptions about the relative difficulty of querying such sources.

[47] http://lod.openlinksw.com/sparql
[48] http://virtuoso.dbtune.org/sparql
[49] http://www.last.fm/api

Name: Irish_Artists_Popularity_on_MySpace

Elements: (Data Type C)

- *Total friends on MySpace is*
- *Country from is*

Type: Hybrid

Rules:

Return Type = MusicArtistName

- **Parameter Name:** Highly
  Return all MusicArtistNames where
  MusicArtistName *Total friends on MySpace is* > 50000
  AND MusicArtistName *Country from is* = Ireland

- **Parameter Name:** Moderately
  Return all MusicArtistNames where
  MusicArtistName *Total friends on MySpace is* ≤ 50000
  AND MusicArtistName *Total friends on MySpace is* ≥ 5000
  AND MusicArtistName *Country from is* = Ireland

- **Parameter Name:** Lowly
  Return all MusicArtistNames where
  MusicArtistName *Total friends on MySpace is* < 5000
  AND MusicArtistName *Country from is* = Ireland

Figure 5-6. Sample Semantic Attribute for SABer Evaluation

The participants were given the rules to encode into semantic attributes so there was no need for them to be "experts" in the domain. This was justified as it was the ease in which coherent rules could be constructed by non-technical users that was being evaluated rather than the usefulness of the semantic attributes created. This chapter has already described how useful semantic attributes can be deployed in various domains and applications, so it was not necessary for this to be evaluated in this experiment.

During the course of the experiment the length of time it took each semantic attribute to be created was recorded, the accuracy of the finished semantic attribute noted (whether it exactly matched the semantic attribute given to the participants on paper) and any questions or problems that arose. Users were also given the opportunity to create semantic attributes of their own after creating the nine semantic attributes set for them. If users could successfully generate their own semantic attributes, it would help to demonstrate how users can independently use SABer to encode SME after a minimal training period.

Once the users were finished using SABer, each completed a short post questionnaire (see Appendix F and an SUS test (System Usability Scale, see Appendix G) (Brooke 1996). Previous studies have shown that summative usability studies using a SUS test should have sample sizes of at least 12 participants (Lewis & Sauro 2009), hence it was possible to get an overall SUS score for technical and non-technical groups as well as an aggregate value for all 24 participants. The SUS test provided an indicator of the usability of SABer and the post questionnaire gave space for participants to elaborate on any usability issues or functionality they would like to see. Moreover, the post questionnaire asked participants to specify if they found inputting rules for any of *expert*, *hybrid* or *template* semantic attributes considerably more difficult, and likewise if they found inputting rules for any of query type A, B or C significantly more challenging. These questions helped assess if part of the SABer application needed to be adjusted to make inputting semantic attributes of certain types more intuitive. Moreover they helped determine if the Semantic Attribute Model was sufficiently generic and abstract to allow users to ignore the underlying idiosyncrasies of querying different data types.

Once the experiment was completed the results from the twelve technical users were used to provide a baseline performance in terms of speed and accuracy of creating the semantic attributes. In previous evaluations it had been shown that technical users can use SABer to generate useful and deployable semantic attributes, thus comparing the time it takes for non-technical users to create accurate semantic attributes with that of technical users, it would give a good indicator as to the usability of the tool. If the group of non-technical users performed, on average, at a level of accuracy and efficiency close to that of their more technical counterparts, it could be reasonably concluded that SABer was suitable for non-technical domain experts to use. Moreover, by comparing the average SUS score for the technical, non-technical and overall groups, a good indication of the tool's usability would be garnered.

5.6.3 Experimental Results

5.6.3.1 Effectiveness of SABER

As outlined previously the effectiveness of SABer in allowing non-technical users to generate semantic attributes was measured by comparing their average speed and accuracy with the averages of their technical counterparts. With regards to the speed in creating semantic attributes, Figure 5-9 shows the average time in seconds it took for each of the nine semantic attributes to be created (with error bars showing the Standard Error). On average the non-technical users were only 8.4% slower than their technical counterparts.

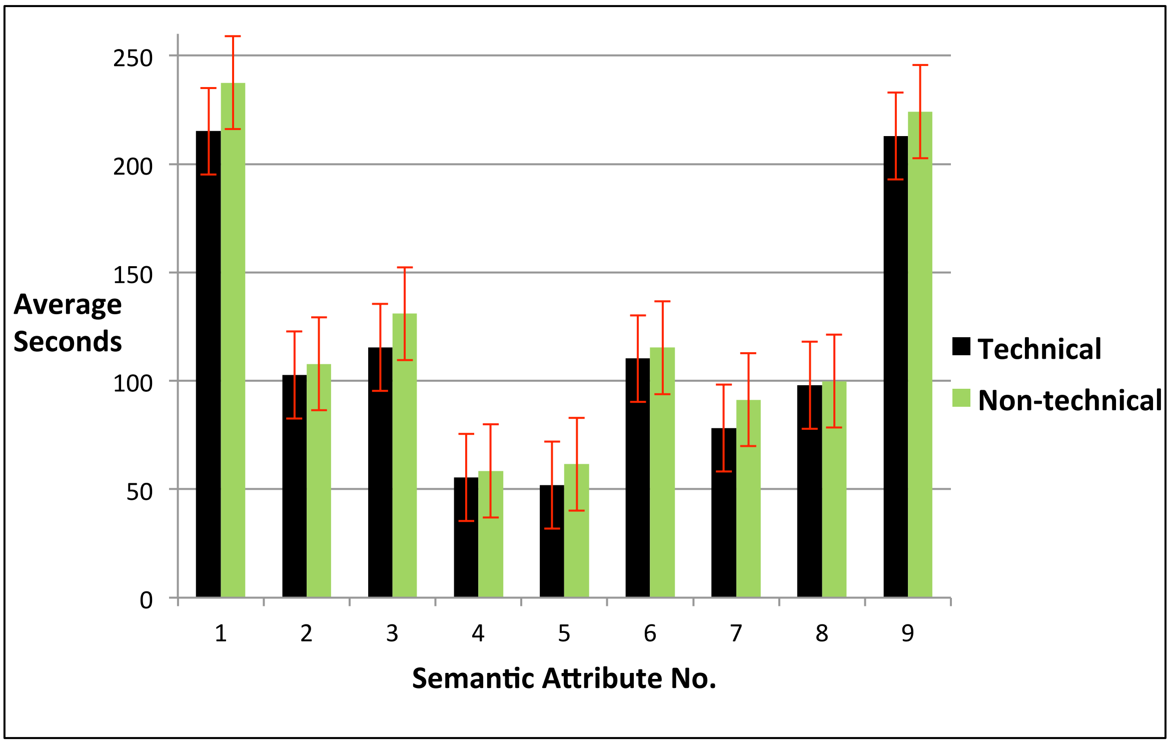


Figure 5-7. Average Time Technical and Non-Technical Users Took to Create Each Semantic Attribute

The accuracy of the created semantic attributes was determined by comparing the Semantic Attribute Model generated by SABer to the data specified to be inputted in each task sheet. In terms of the accuracy of the semantic attributes created, the difference in performance was even smaller between the groups than in the speed comparison. Technical users on average got 8.5 out of 9 accurate (SD 0.67) with non-technical users getting 8.2 of 9 accurate (SD of 0.84). Figure 5-10 displays these details.

It must also be noted that all inaccuracies by participants in both groups can be classified as “slips” where wrong numeric values or misspelt words were inputted by the users. Slips are defined by attentional failures where the action was unintended (Reason 1990). These kinds of errors can be easily corrected, and there were no cases of a user fundamentally not being able to create a semantic attribute or giving up half way through. Furthermore, all users were able to create their own semantic attributes after completing the nine semantic attributes that were set for them. Many of these semantic attributes were of comparable complexity to those set for them during the experiment. This demonstrated that users could independently create semantic attributes in SABer after a minimal amount of training.

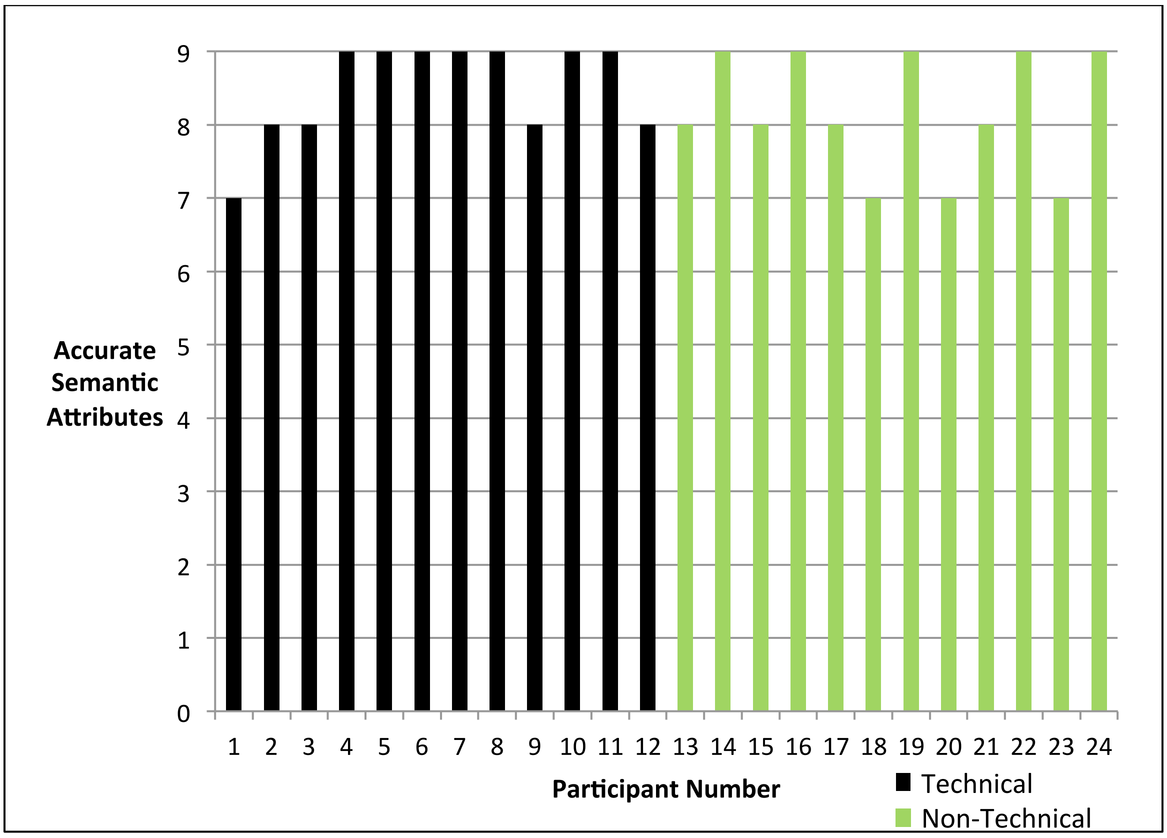


Figure 5-8. Number of Accurate Semantic Attributes Generated by Technical and Non-Technical Users

5.6.3.2 Usability of SABer

Figure 5-11 shows the SUS scores for the authoring tool which help determine the system's usability. On average the technical users (participants 1-12) gave SABer a SUS score of 83.3% (SD 9.4), and the non-technical users (participants 13-24) 74.4% (SD 10.7) The average SUS score for all 24 users was 78.85% (SD 10.05). Systems that score above 72.5% on the SUS scale can be classified as having good usability (Bangor et al. 2009), so it can be concluded that SABer is considered a usable tool by both non-technical and technical users.

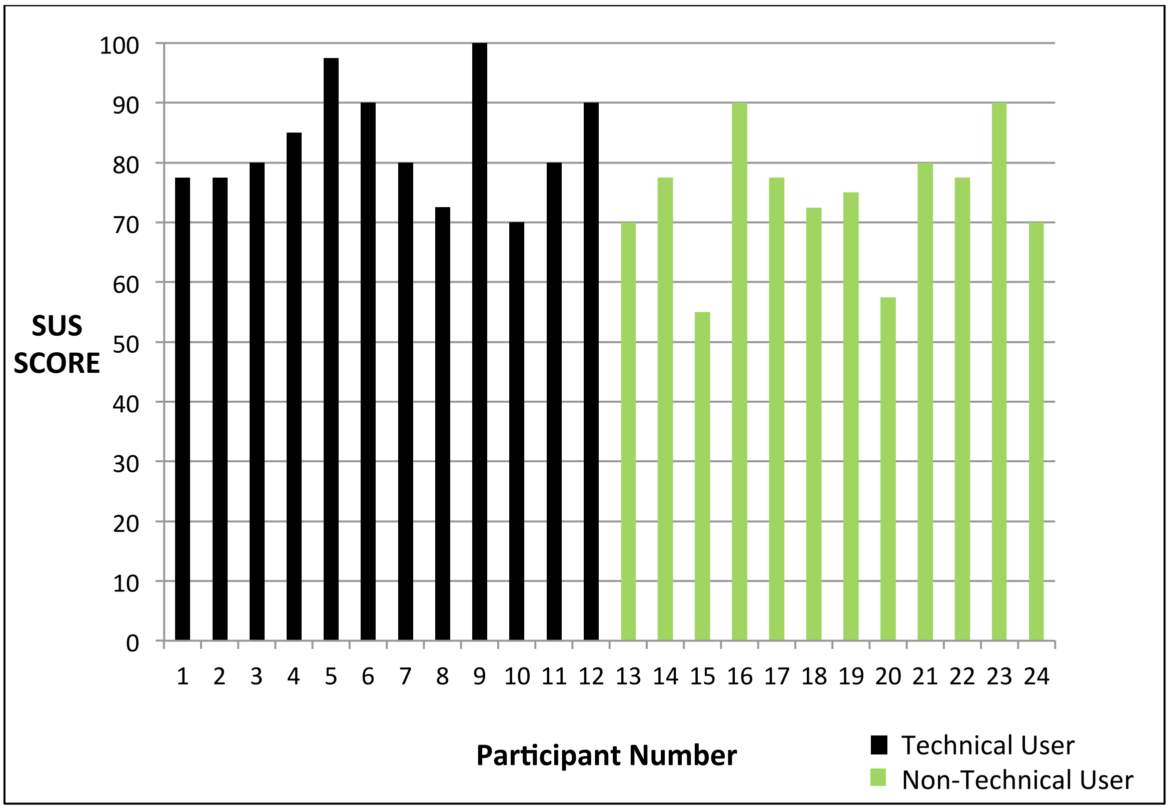


Figure 5-9. SUS Scores Given by Technical and Non-Technical Users

During the post questionnaire there was an opportunity for participants to make interface suggestions and request functionality. The only major interface suggestion that was mentioned frequently was to change how the metadata elements were presented in step one. It was felt that it would be easier to choose the elements to be joined if they could be sorted alphabetically as well as by superclass, source and collection (for XML type sources). Some minor suggestions regarding spacing and font sizes were also mentioned sporadically but overall users were largely satisfied with the interface, especially in how it facilitated rules to be generated. In terms of the functionality that users would like to see in SABer, the only significant ones mentioned were the ability to load and edit saved semantic attributes, and to be able to join them into a compound semantic attribute if appropriate. These compound semantic attributes could then be tested for results and labelled for deployment in SARA if useful. Both these functionalities were not necessary for this particular experiment, however it is planned for them to be included in future releases of SABer.

5.6.3.3 Supporting Different Semantic Attribute and Query Types

The post questionnaire that participants filled in asked them to specify if they found inputting rules for any of *expert*, *hybrid* or *template* semantic attributes a considerably more difficult challenge. Figure 5-12 shows how 22 of the 24 users found no significant extra difficulty in creating semantic attributes of different types (*hybrid, template or expert*). None of the participants felt that creating *template* semantic attributes was more difficult than any of the others. This meant that all participants were comfortable with selecting "some text" or "some number" while creating rules instead of inputting specific values. This was important to validate as non-technical users would typically not be as familiar with the concept of a variable as technical users would, and this concept had to be presented to them in an intuitive fashion.

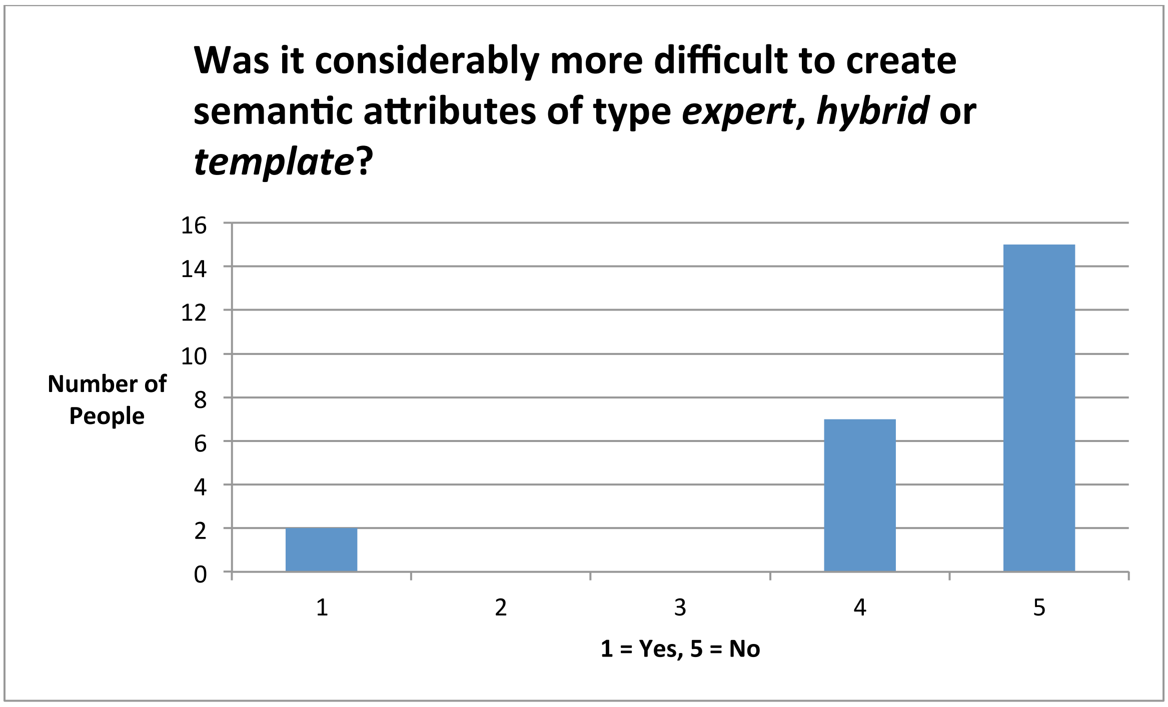


Figure 5-10. No. Participants Who Found it Considerably More Difficult to Create Semantic Attributes of Type *expert*, *hybrid* or *template*

The post questionnaire also asked if participants found inputting rules in one of query type A, B or C significantly more challenging. Figure 5-13 shows that 22 out of the 24 users found no significant difference in difficulty in creating semantic attributes using XQuery (Query type A), SPARQL (Query type B) or API calls (Query type C). This was important as it showed that users were largely indifferent to the underlying technologies they were working with, and meant that the semantic attribute was sufficiently abstract to distance these users from the underlying technical complexity of each query language.

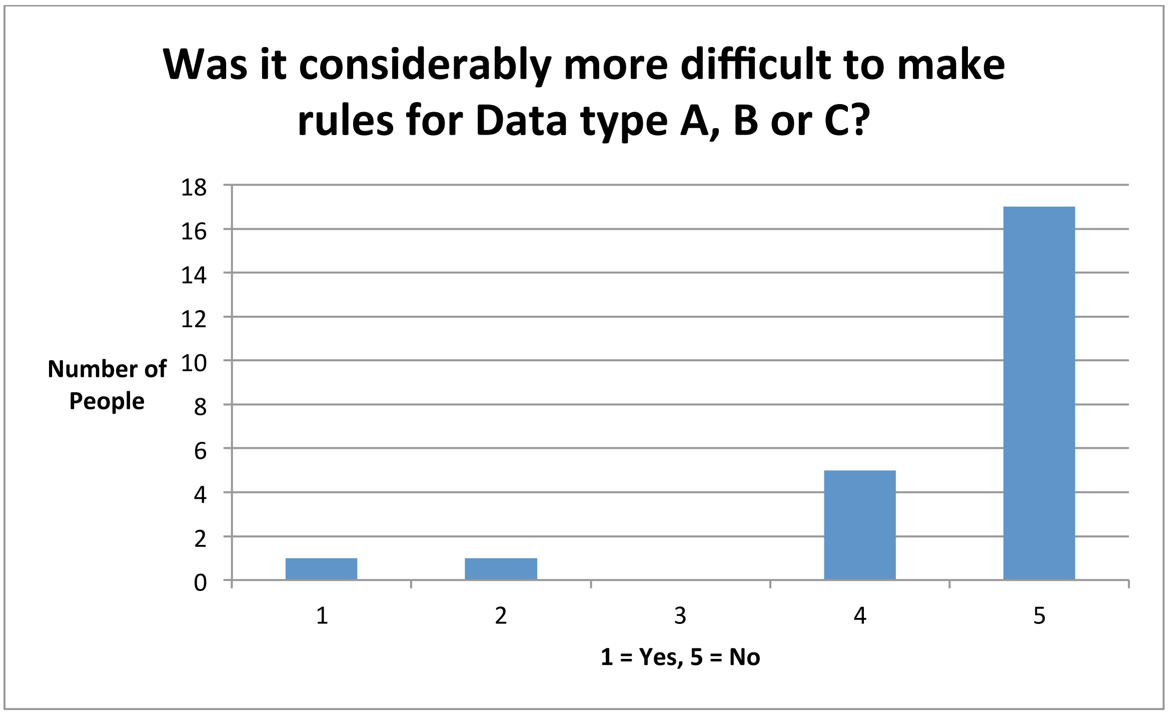


Figure 5-11. No. Participants Who Found it Considerably More Difficult to Make Rules for Data type A, B or C?

5.6.3.4 Summary

Usability is defined by the International Organization for Standardization (ISO) as the effectiveness, efficiency and satisfaction with which a specified set of users can achieve a specified set of tasks in a particular environment (Jokela et al. 2003). Participants in both the technical and non-technical groups were shown to perform the tasks effectively and efficiently. Moreover, there was no significant difference in speed, accuracy or perceived usability of SABer between the two groups. Users also found no significant difference in creating semantic attributes of different types or with different query languages, thus showing that SABer was sufficiently abstract to distance users away from the underlying complexity of the data sources. It can thus be concluded from this section that SABer is a tool with good usability and that domain experts without a computer science background could use it to create semantic attributes with a minimal amount of training. This highlights how feature E2 from Section 5.1 was fulfilled by SABer, and shows that it should be possible for experts in any metadata rich domain to capture SME as semantic attributes if given a sufficient choice of elements. The functionality that SABer offers domain experts is rooted in the design requirements of the framework associated with the *expert-supported approach to data exploration*. This close coupling of design and implementation meant that SABer's successful evaluation also helped to validate the underlying approach and framework.

5.7 Experiment 5: SARA Performance Evaluation

5.7.1 Experimental Goals

SARA was implemented as a prototype system that embodied the *expert-supported approach to data exploration* that was outlined in chapter 3. As has already been highlighted in this chapter, this prototype middleware has been successfully used with different applications in various domains. However, as a middleware system it was necessary to quantify how much overhead SARA added in terms of overall query latencies between client applications and data sources. A middleware system should be as efficient as is possible, thus by quantifying its overhead it helps show under what circumstances the current prototype of SARA is a usable system. By detailing such overheads it also makes clear whether these costs are outweighed by the benefits that SARA's features offer client applications. Thus this experiment aims to satisfy the procedure P1 outlined in Section 5.1. It must be stressed that the particular implementation of SARA evaluated is only a prototype system, hence this performance evaluation focussed on its core feature (processing Semantic Attribute Queries sent from a single client application). The three main aims of this experiment were thus:

- Demonstrate that the speed in generating Result Models of the same size was independent of the type, location and size of source being accessed.
- Show that processing time within SARA, apart from that taken to generate Result Models, does not fluctuate hugely or take up a significant amount of time (in the context of the typical overall latencies experienced by client applications who send queries and process results – often measuring in seconds - for this experiment, lengths of time less than 5ms were deemed insignificant).
- Examine how quickly SARA can generate Result Models of various sizes.

5.7.2 Experimental Setup

SARA is a middleware system and so does not store data directly, but rather proxies queries to individual sources and reconciles the results. Hence the length of time for a query to be returned to a client application is very dependent on the location and type of data being accessed. For instance, issues such as the network quality between SARA and the different data sources, the load on these repositories when the queries are being sent, and the size and optimisation of these individual sources, will have a huge effect on query latencies. As these factors vary hugely and are independent of SARA itself, this experiment focuses on measuring the overhead of the query that can be attributed to operations within SARA. The length of time that data sources spend processing queries is unaffected whether they are queried directly by a client application without using SARA, so it is the overhead added by SARA that is of most relevance to this thesis.

Three different data sources for this experiment were selected as representative instances of the types of data sources that SARA can access and send queries to. The three sources were selected because they contained data in XML, data in RDF and data accessible through a Web API respectively, as well as having a similar (or larger size) than the datasets previously accessed by client applications of SARA. They were as follows:

- A local XML database containing 21,755 songs that charted in the US Billboard charts. This consisted of a 435,013 line XML file (17.7MB) stored in an eXist 1.2.4 database, running on an Apache Tomcat Servlet Engine (version 6.0.14e).
- Last.fm Web Services containing information on 280,000 artists and labels, and over 7 million tracks[50]
- The http://lod.openlinksw.com/ SPARQL endpoint which contains over 7.59 Billion Triples[51] from the Web of Data

This experiment was performed on a Dell Precision M2400 Laptop with an Intel Core 2 Duo Processor (2.6GHz) and 4GB of RAM. The Operating System installed was a 32 Bit version of Windows 7, and the Semantic Attribute Queries were sent to SARA via an application developed in Eclipse 3.5.2. An insignificant overhead is introduced as a result of using Eclipse, and this time is accounted for as part of the Miscellaneous SARA Time described in Table 5-4. All times for each operation were logged to a millisecond precision within Eclipse. The times associated with SARA's processing would of course decrease considerably if a more powerful computer or sophisticated parallel processing techniques were used. However, this experiment aimed to highlight the prototype's speed on a modest machine not optimised for performance. Table 5-4 describes the various operation processing times recorded for this experiment.

[50] http://blog.last.fm/2009/03/24/lastfm-radio-announcement
[51] http://virtuoso.openlinksw.com/presentations/Linked_Data_Virtualization/Linked_Data_Virtualization.ppt

Table 5-4. Various Operation Processing Times Recorded for the Performance Evaluation of SARA

Overall Execution Time	The total time from when a query is sent by the application, to when the Result Model is presented onscreen.
Query Execution Time	The time taken for the results to be returned from the data source for all queries sent by SARA. Note that if a Semantic Attribute Query contains more than one semantic attribute then these queries are currently sent one after the other, with the next query not firing until results from the previous query have been sent back.
Result Model Generation Time	The time it takes to generate the final XML Result Model.
Miscellaneous SARA Time	Any other time taken within SARA such as decomposing Semantic Attribute Queries into individual queries, or reconciling multiple result sets. It also includes the minor additional time spent calculating the various times in Eclipse which is not necessary for released versions of SARA.

This experiment was divided into three parts. First, queries encapsulated within semantic attributes were sent to the eXist database, to the Last.fm Web Service, and the SPARQL endpoint, in order to demonstrate that the speed in generating Result Models of the same size was independent of the type, location and size of source being accessed. Three Semantic Attribute Queries were sent to each source, with each query returning approximately 50, 150 and 250 results. Because these queries returned result sets of a comparable size, it helped to determine if results sets from data sources of a particular format took longer to process by SARA than others. Each of these queries was sent ten times and an average length of time to generate the Result Model was stored for each one. This allowed comparisons to be made regarding the time spent generating Result Models of a similar size. Table 5-5 shows the specific queries sent from each semantic attribute, the type of source they were accessing, and the number of results that the queries returned.

Table 5-5. List of Queries , Source Types and No. Results used in Part One of the Performance Evaluation of SARA

Source Type	**Queries**	**No. Results**
eXist DB	Rock Artists with the top 86 US Singles of 1991	50
Web API	50 most similar artists to "Ben Folds"	50
SPARQL endpoint	All songs by "My Bloody Valentine"	59
eXist DB	Rock Artists with the top 140 US Singles of 1970 or 1972	150
Web API	150 most similar artists to "Van Morrison"	150
SPARQL endpoint	All songs by "Therapy?"	154
eXist DB	Rock Artists with the top 160 US Singles of 1981, 1981, 1983 or 1984	248
Web API	250 songs most similar to U2's "One"	250
SPARQL endpoint	All artists with a song called "Today"	257

The second part of the experiment involved Semantic Attributes Queries that were sent to SARA (some with single and some with multiple semantic attributes), in order to demonstrate that the processing time within SARA, apart from that taken to generate Result Models, does not fluctuate hugely or take up a significant amount of time. As this part of the experiment was focussed on measuring internal processing times of SARA, regardless of what source the results were coming from (which was examined in the first part of this experiment), the queries run were only required to return results sets of various sizes, and each query was run ten times to get an average duration. All the times specified in Table 5-4 were stored (Overall Execution Time, Query Execution Time, Result Model Generation Time and Miscellaneous SARA Time), so that the significance of each process, in terms of time, could be analysed. The various Semantic Attribute Queries that were sent in this part of the experiment were:

- Top Singles in US Charts in 1990s
- Number Very Low Tempo Songs Charting in US
- Number Very High Tempo Songs Charting in US
- Rock Artists with songs in the top 50 selling of a year, in the 60s, 70s, 80s or 90s. (four separate semantic attributes combined)
- Rock Artists with songs in the top 200 selling of the year, in the 60s, 70s, 80s or 90s. (four separate semantic attributes combined)

The final part of the experiment aimed at testing how quickly SARA could generate Result Models of various sizes, thus it was necessary to send various queries that returned increasing amounts of results. Hence, 15 Semantic Attribute Queries (each containing a single semantic attribute relating to the tempo of songs in the eXist database) were sent to SARA, with each returning a result set ranging in size from 53 to 18,177. Each of these queries was sent ten times and an average latency for the Result Model generation was calculated.

Specifically this experiment looked at measuring the length of time it takes to generate a standard Result Model from various Semantic Attribute Queries that do not require superclass conversion. The reason for focussing on Semantic Attribute Queries that do not require superclass conversion, is that this conversion functionality requires a minimum of one further query to be sent for each result (i.e. sending a query to convert from an Album ID to an Artist ID). Hence, if there are a large amount of results from the original query sent by the client application, it means that this initial query may spawn many hundreds or thousands of extra queries. The speed of these queries is very dependent on the particular data source being accessed and the network conditions at that time. As mentioned in Chapter 4, the current prototype of SARA sends these extra queries one at a time, with the next query firing after the results from the previous query have been returned. This was as a result of a decision to focus on implementing the core functionality of SARA rather than ensuring that all its operations would perform to a production level performance. Hence, the SARA prototype is only suitable for handling superclass conversions if there are a small amount of results, and the application does not view these queries as time critical.

5.7.3 Experimental Results

5.7.3.1 Comparison of Result Model Generation Speed for Different Data Sources

This thesis hypothesises that the speed of generating a Result Model by SARA is independent of the data source type. This is because the Result Model is generated from a HashSet of identifiers reconciled from the separate data source results. Creating a Result Model involves converting this HashSet into an XML file that can be parsed by the client application. Hence, the number of results in the HashSet (and typically to a lesser extent the number of characters in each result identifier) is the key factor in determining the length of time it takes to generate the Result Model.

In terms of all the queries listed in Table 5-5, the average length of time taken for all queries listed to generate a Result Model was less than 1ms. Though the length of time taken to generate the larger Result Models in this table was likely to have been longer (though still sub-millisecond), it was not possible to measure this difference as millisecond granularity was the level of precision recorded for this experiment. However, the standard deviation for every query run was zero (at millisecond granularity), which demonstrates that SARA consistently generates Result Models of the same size at approximately the same speed, regardless of whether results come from a data source located locally or remotely, or if accessed via XQuery, SPARQL or through a native API. This supports the hypothesis that the performance of SARA in generating a Result Model is independent of the size, location or type of data sources that have been queried.

5.7.3.2 Comparison of Processing Speeds for Different Operations within SARA

This thesis hypothesises that the processing time within SARA, apart from that taken to generate Result Models, does not fluctuate hugely or take up a significant amount of time. As mentioned earlier, for the purposes of this experiment, times less than 5ms were defined as insignificant. The total height of each column in Figure 5-14 depicts the total length of time for a semantic attribute (Top Singles in the US Charts in the 1990s) to be sent to the eXist database and its 100 results to be processed by SARA. Each column (representing this total time) is divided into three, representing (from bottom up) the Query Time, the Result Model Time (which does not feature in this case because it never averaged over 1ms) and the Miscellaneous SARA Time. The query was sent ten times, and Figure 5-14 shows how the Miscellaneous SARA Time stays very consistent.

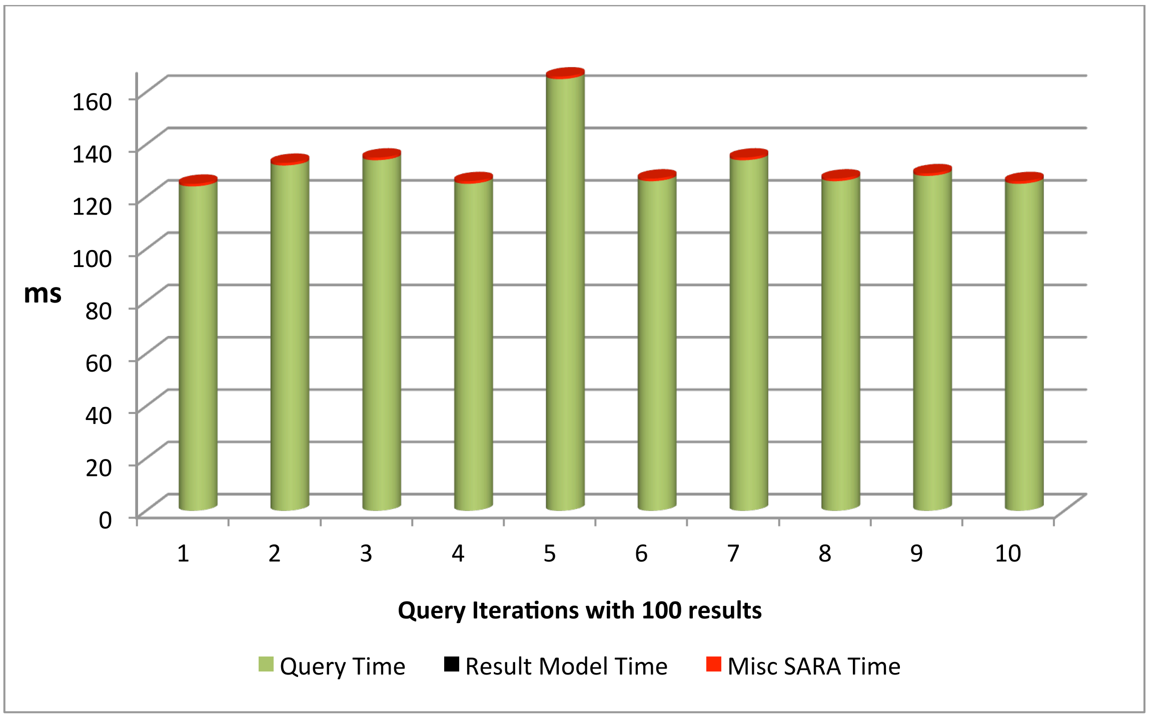


Figure 5-12. Time Spent Processing Semantic Attribute "Top 100 Singles in US Charts in 1990s"

Figures 5-15, 5-16 and 5-17, show three further semantic attributes that were each sent ten times to the eXist database and processed by SARA. The results are plotted in the same manner as just described in Figure 5-14. These semantic attributes returned larger result sets (444, 984 and 11,448 respectively), with the increased processing time SARA needed to generate the Result Model clearly visible in figures 5-16 and 5-17 (note that Y axis in Figure 5-17 starts at 540 to make it easier to compare with previous figures). Despite the increased number of results, the Miscellaneous SARA Time is negligible compared to the length of the overall query (fluctuating between 1ms and 5ms).

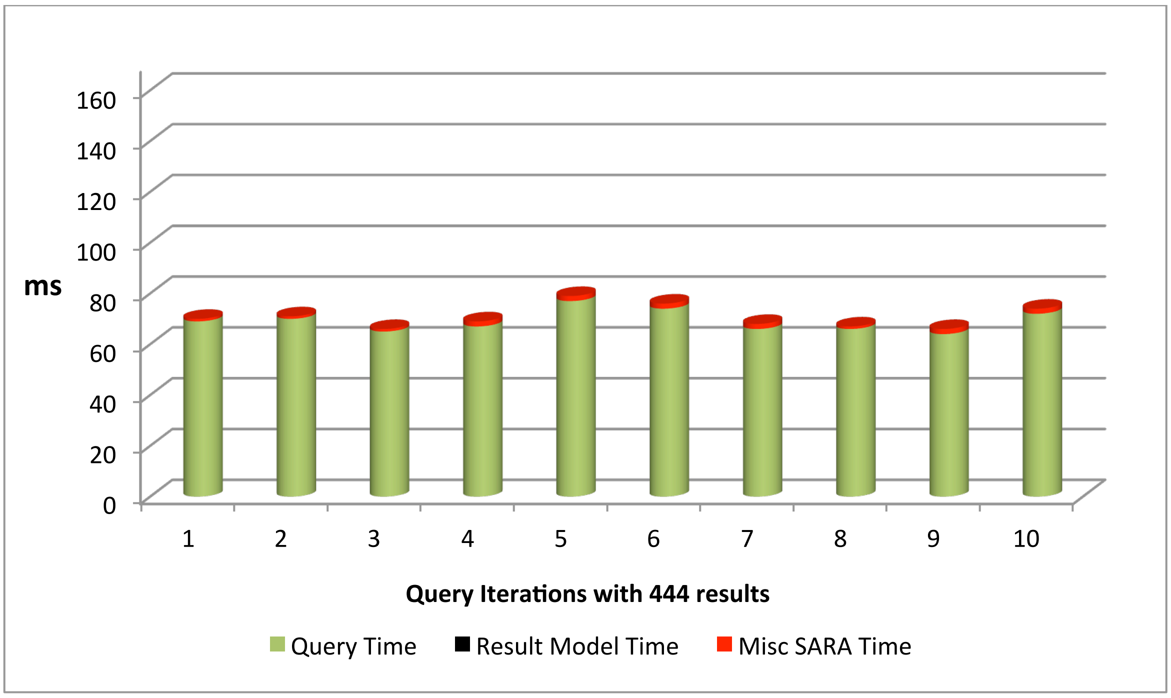


Figure 5-13. Time Spent Processing Semantic Attribute “No. Very Low Tempo Songs Charting in US”

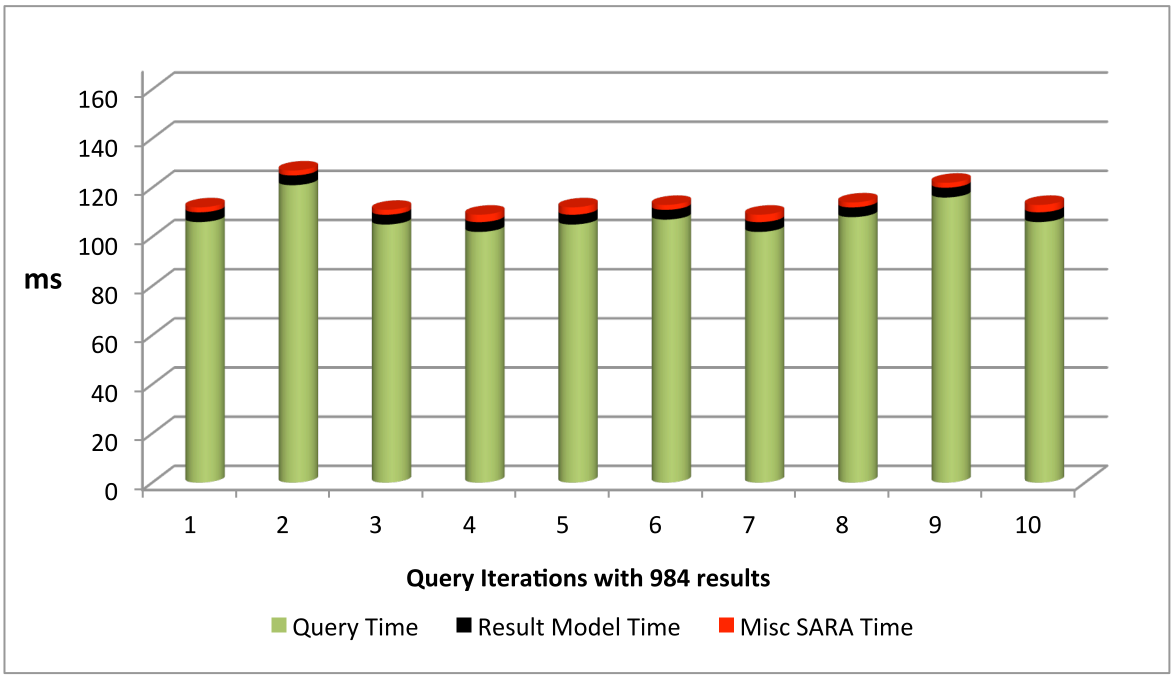


Figure 5-14. Time Spent Processing Semantic Attribute “No. Very High Tempo Songs Charting in US”

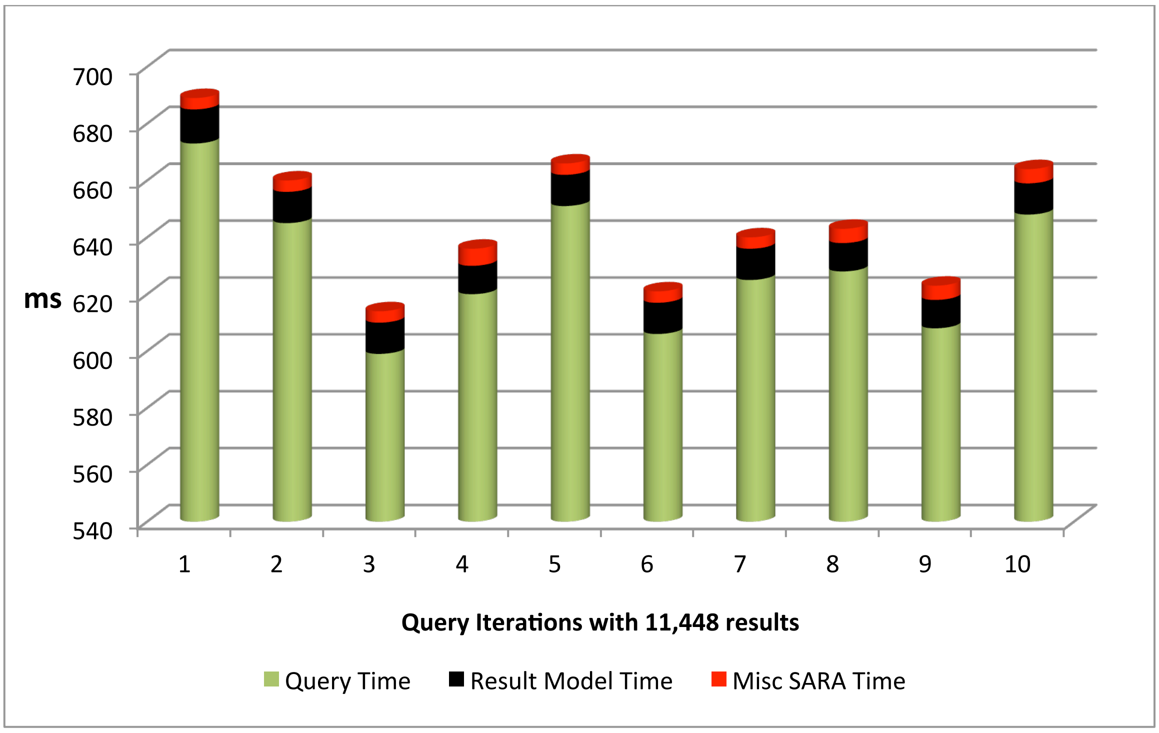


Figure 5-15. Time Spent Processing Semantic Attribute "No. Average Tempo Songs Charting in US"

The lack of time spent processing in SARA is further highlighted in Figures 5-18 and 5-19 where more complex Semantic Attribute Queries are sent. These queries consist of four separate semantic attributes joined together that returned result sets of 794 and 1820 respectively. For Figure 5-18 the miscellaneous SARA time only averaged 1ms and for Figure 5-19 it averaged 2ms. Please note that Figure 5-18 and Figure 5-19 start their Y-axis at 540 and 840 respectively to make them easier to compare.

Another interesting feature of Figures 5-18 and 5-19 is the length of each query time. As discussed earlier, SARA processes Semantic Attribute Queries that contain multiple semantic attributes in sequence. This means that the query encapsulated in each of the four semantic attribute was sent in turn, with the next one fired once results had been received by SARA for the previous query. In future versions of SARA, when these queries are sent in parallel using multi-threading, then the query time component should reduce significantly, thus bringing down the overall query time experienced by the client application.

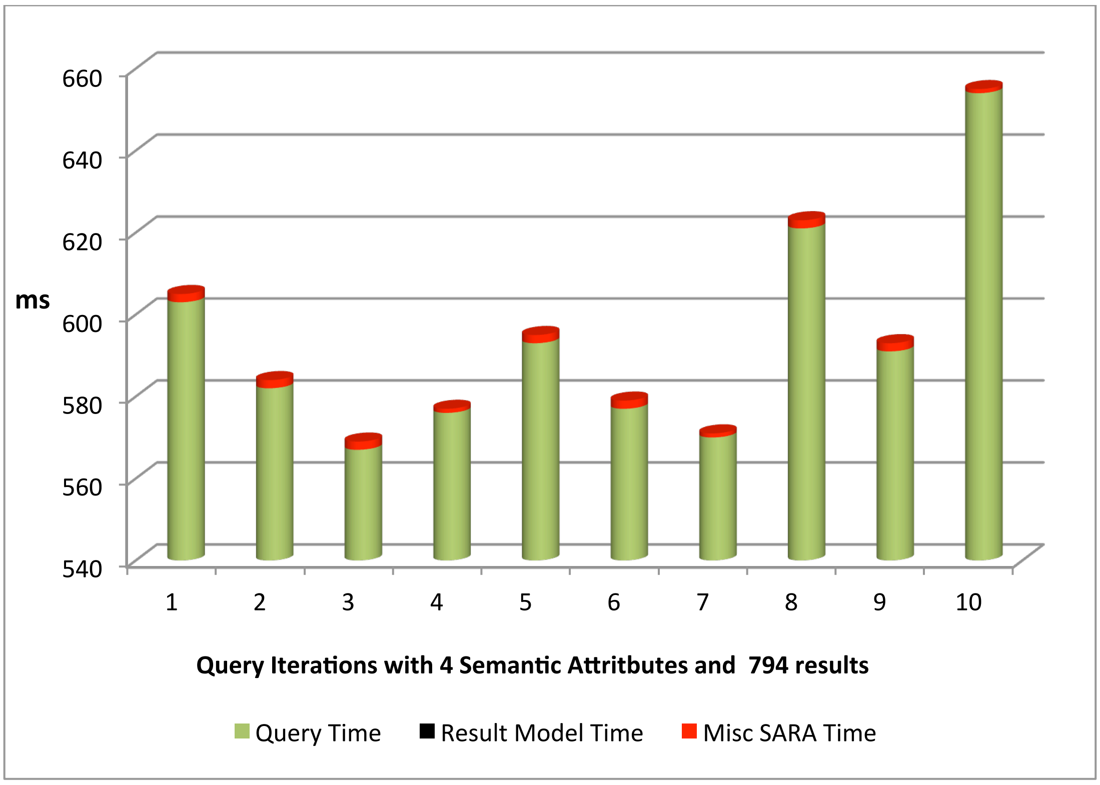


Figure 5-16. Time Spent Processing Semantic Attribute Query "Rock Artists with Songs in the Top 50 Selling of a Year, in the 60s, 70s, 80s or 90s"

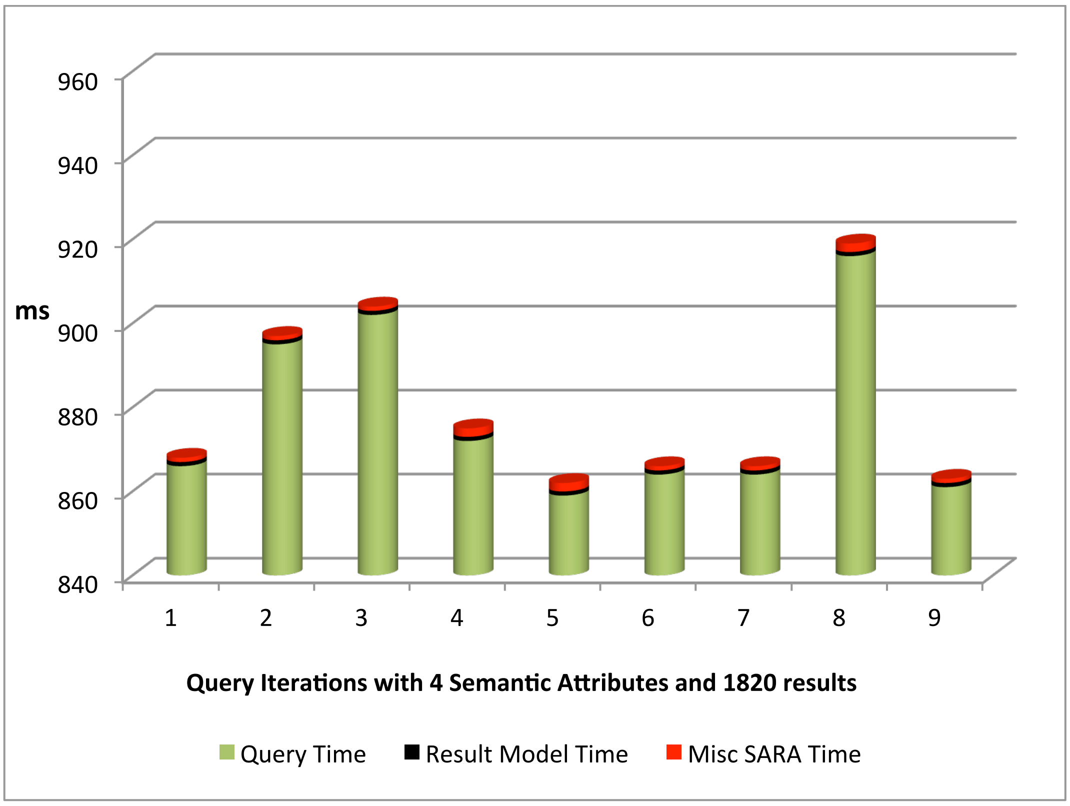


Figure 5-17. Time Spent Processing Semantic Attribute Query "Rock Artists with Songs in the Top 200 Selling of a Year, in the 60s, 70s, 80s or 90s"

These results have demonstrated that all processing time within SARA, apart from the time taken to generate Result Models, does not fluctuate greatly or take up a significant amount of time i.e. less than 5ms._ This was the case even when Semantic Attribute Queries contained multiple semantic attributes which required many results to be reconciled. Surprisingly, it was also noted that the most processor intensive operation of SARA (Result Model generation) was also not a hugely significant proportion of the overall query length, which consisted mainly of time spent querying the data sources themselves. The average time taken for Result Model generation is examined in more detail in the following section.

5.7.3.3 Speed of Result Model Generation in SARA for Variously Sized Result Sets

The final part of this experiment looked at how quickly SARA could generate Result Models of various sizes. As shown in the previous results section, all other processing time in SARA do not fluctuate hugely or take up a significant amount of time, relative to the overall query latency. Table 5-7 shows the average speed (over ten iterations) for generating Result Models of different sizes, ranging from 53 to 18,177 results.

Table 5-6. Average Length of Time SARA takes to Generate Result Models of Different Sizes

No. Results	Avg Result Model Generation Time in ms	Standard Deviation
53	0	0.00
546	0.1	0.32
666	4.2	0.42
984	4.9	1.52
1,697	1.1	0.32
2,359	1.2	0.42
3,380	2.0	0.00
3,953	2.0	0.00
4,952	3.0	0.00
5,601	3.1	0.32
5,903	8.2	0.42
6,234	8.0	0.00
11,727	10.6	0.70
16,363	14.6	1.71
18,177	15.2	0.42

Figure 5-20 visualises the data from Table 5-7 as a graph. The overall trend of the graph is as expected, with the length of time generally increasing as the number of results to be processed increase. As shown in the graph a large Result Model of over 18,000 results can be processed in just over 15ms, so in the overall context of query latencies, this overhead is unlikely to have a noticeably negative impact on a user's interaction with a client application.

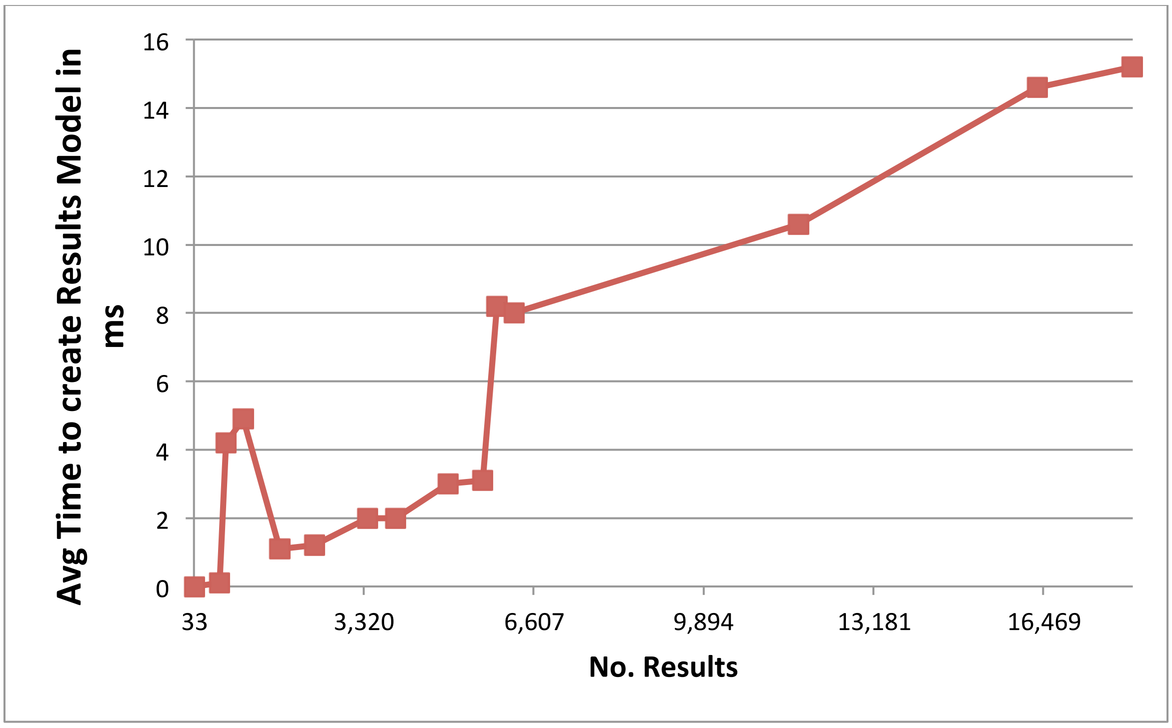


Figure 5-18. Total Result Model Size Graphed Against the Average No. Results Added per ms.

5.7.4 Summary

The main aim of this experiment was to examine the speed at which SARA generates Result Models of various sizes, in order to show under what circumstances the current prototype of SARA is a usable system. This satisfied feature P1 outlined in Section 5.1. Before tackling this task, the first two parts of the experiment involved demonstrating that the speed of generating Result Models of the same size is comparable regardless of the type, location and size of source being accessed, and that all processing time within SARA, apart from Result Model generation, does not fluctuate hugely or take up a significant amount of time relevant to the overall query. Demonstrating these features was important as it showed that application developers can access repositories with complex queries, even with billions of triples (see Section 5.7.2), without any significant penalties in terms of overall query latencies.

Overall, the performance of the SARA prototype on a modest computer is likely to be acceptable to many potential applications (especially those wishing to access multiple remote sources), as a typical query returning over 18,000 results can be processed in approximately 15ms. To put this in context, the Film Domain Exploration Client only required 50 results to be returned at a time, and a typical X2Photo query returned result sets in the hundreds. Though time taken to generate Result Models does change as expected depending on the number of results that need to be processed, in the context of queries that return less than 5,000 results (these add less than 5ms to the overall query latency) the fluctuation is minimal and unlikely to impact on user satisfaction with the client application. Furthermore, it must be reiterated that this prototype version of SARA was evaluated on a modest laptop not optimised for performance, and that the length of time associated with SARA's processing would decrease considerably by using a more powerful computer or more sophisticated parallel processing techniques.

The main issue of concern for potential application developers, in terms of the current SARA prototype, should be whether the corresponding overhead in SARA will affect their user's interaction with their application. The results of this experiment provides them with a guideline of what performance they can expect, and allow them to better judge whether the associated overhead of SARA is outweighed by its features. As mentioned previously, SARA's performance is likely to be acceptable to many types of applications. However, it should be noted that even when the performance of SARA is extremely efficient, the slow speed of processing within individual data repositories or bad congestion in a network, may result in the overall performance not being sufficient for particular applications. This issue is common with applications reliant on accessing remote sources, and advances in this research area will benefit all distributed systems in general.

5.8 Analysis of Evaluation Results

The aims of this chapter were to show how feedback from preliminary evaluations helped shape the design and implementations of both SARA and SABer, as well as to provide further evidence that the following four features were being supported:

E1 SARA provided client applications access to data sources without prescribing a specific user interaction paradigm for the GUI, or for their developers to know the underlying query language associated with each source.

E2 SABer enabled the SME leveraged by the client applications to be encoded by non-technical experts.

E3 Casual users that appropriate (or tailor) this SME within a client application could send complex queries to data sources via SARA.

E4 SARA and SABer are domain independent.

Furthermore, as middleware systems can cause an extra bottleneck that client applications must pass data through, it was also necessary to test SARA's performance in order to help quantify to what degree it increases query latencies. This procedure was summarised as:

P1 Performance evaluation of SARA to see under what circumstances it is a usable middleware

Figure 5-21 depicts the hierarchy that exist from this thesis' research question down to the individual evaluation experiments (please note that the size of the individual boxes is not an indicator of their relative importance). It shows how the evaluation features that were looked for in the various experiments stemmed directly from research objectives, and ultimately the research question. As mentioned previously, the case study in Section 4.4 describes how SARA accommodates the features encapsulated in C1 (*technical features of SARA showcased in music case study*), so this chapter did not reiterate these findings.

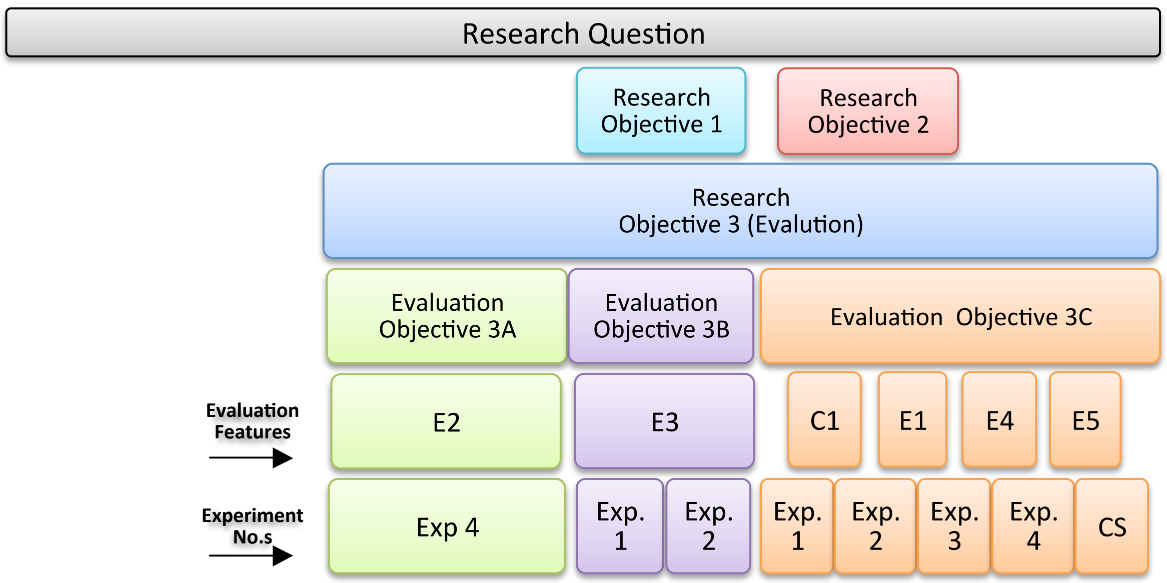


Figure 5-19. Hierarchy from Research Question down to Individual Evaluation Experiments

The experiments described in this chapter showed how the various evaluation features were present in SARA or SABer, and that the evaluation objectives that they were based on were attained. An analysis of the evaluation experiments is now presented from the perspective of three main stakeholders (the end user, domain expert, and the application developer), and is followed by a comparison of SARA and SABer to the state of the art.

5.8.1 Stakeholder Perspective

The main aim of SARA is to support data exploration by casual users, hence it was important that Experiment 1 and Experiment 3 explicitly showed that users found features, which SARA directly facilitated, in the application as useful e.g. making queries over multiple data sources in a consolidated fashion, using GUIs not restricted to a specific interaction paradigm, and tailoring semantic attributes to their own interpretation. These experiments showed that casual users can benefit from leveraging expertise while exploring a domain of interest to them, which was vital in demonstrating the real potential for larger systems built upon these technologies and their underlying approach.

From the domain expert's perspective it was possible to see the evolution of SME encoding throughout the various experiments. In Experiment 1 (Section 5.3) there was no tool at all and the SME had to be encoded manually, but in Experiment 2 (Section 5.4) three different domain experts successfully used the prototype version of SABer to generate semantic attributes and provided feedback on refinements that could be made. Experiment Three (Section 5.5) detailed how semantic attributes created in SABer were deployed in different applications, and successfully utilised by end users both explicitly and implicitly. Finally, Experiment 4 (Section 5.6) showed that the updated version of SABer was a tool with good usability, and that domain experts without a computer science background could use it to create semantic attributes with a minimal amount of training. This was crucial in showing the novelty of the system and its relevance to metadata rich domains. The flexibility of the Semantic Attribute Model, which can define SME in multiple languages was also validated, and again provided further evidence of the uniqueness and wide applicability of the system.

From a developers perspective it was shown how SARA evolved over the various experiments with user feedback resulting in additional features being added over time e.g. compatibility with new remote sources of different types, refinement of SARA's API, development of reusable models etc. Specifically, the use of a parameter based API and the support for reusing semantic attributes and sources models make the approach underpinning SARA an attractive one for developers wishing to create useful data exploration applications efficiently. Furthermore, in the course of this chapter SARA and SABer have been evaluated in four different domains (music, film, digital imaging and academic publications), which strongly highlights the domain independence of these systems. This is significant in making SARA, and the approach it is based on, as applicable to as wide an audience as possible. Finally Experiment 5 (Section 5.7) showed that the current performance of SARA is good, and that its overhead is unlikely to deter developers from employing it with their applications.

5.8.2 State of the Art Comparison

SARA and SABer are the two main components implemented for this thesis, hence there is a necessity to describe the set of features that they offer which distinguish them from other systems from the state of the art. For instance, in terms of the tools that support *Complex Querying by Casual Users over Multiple Sources* (outlined in Section 2.2.8), the main features of SARA that make it distinct are that it:

- supports reconciliation of data from multiple sources in different data formats
- employs SME encoded by non-technical domain experts (in an evaluated tool) to help end users explore information domains
- allows client applications access to data sources without prescribing a specific user interaction paradigm for the GUI, or for their developers to know the underlying query language associated with each source

Table 5-7 summarises the features offered by SARA in comparison to those systems analysed in Chapter 2. It highlights how SARA offers consolidated access to multiple data sources, in formats commonly used on the Web. SARA also supports users through SME that has been encoded in authoring tool that is usable by non-technical domain experts. Finally SARA allows bespoke applications to be built on top of it, which means developers are free to use existing widgets to create a user interface, but if this is too restrictive, they can instead develop their own user interface that is tailored specifically to the users of their application.

Table 5-7. Summary of Systems that Support Complex Querying by Casual Users

System	**Multiple Sources**	**Multiple Data Types**	**Data Formats**	**SME**	**Main Query Interface**	**Supports Application Development**
Search Computing	Yes	Yes	Web Services and Relational DBs	Yes – no authoring tool	Widgets	Yes – Widget fronted search application
Parallax	No	No	Freebase Knowledge Base	No – auto generated facets	Facets	No
Sparallax	No	No	RDF	No – auto generated facets	Facets	Yes, but only by specifying an endpoint for their GUI
Explorator	Yes	No	RDF	No – auto generated facets	Facets & Query By Example	Yes, but only by specifying an endpoint for their GUI
PowerAqua	Yes	No	RDF	No	NLI	No
Orakel	No	Yes	OWL, F-Logic	Yes – has authoring tool (Framemapper)	NLI	Yes – can be integrated into bespoke application
MashArt	Yes	Yes	Web Services, Atom feeds, JSON, XML	Yes - has authoring tool (MashArt Editor)	Mashup Widgets	Yes - Mashup
SWP	Yes	Yes	RDF, Excel, Relational DB, Text files	Yes – has authoring tool (Ontology Management Component)	Mashup Widgets	Yes – Bespoke Portals
SARA	Yes	Yes	XML, RDF, Web APIs	Yes- has authoring tool (SABer)	Bespoke GUIs	Yes – Bespoke Applications

In terms of the *Domain Independent Tools to Support SME Encoding by Non-Technical Experts* (Section 2.3.7), SABer is novel as it:

- generates SME in different data formats and for use over multiple data sources
- supports tailoring of the rules encapsulated in this SME
- creates a model of SME that is reusable in different installations
- has been evaluated successfully with non-technical users

Table 5-8 summarises the features offered by SABer in comparison to those systems referenced during the state of the art analysis.

Table 5-8. Summary of Domain Independent Tools to Support Non-Technical Domain Experts to Encode SME

Software	SME Generated	Multiple Data Types	Tailorable SME by end user	Evaluated by Non-Technical Users	Reusable	Multiple Sources
Konduit VQB	SPARQL queries	No	No	No	Yes	Yes
SPARQLViz	SPARQL queries	No	No	No	Yes	No
Potluck	Mashup up of Exhibit based sources	No	No	Yes	No	Yes
SpreadATOR	Spreadsheet based Mashup	Yes – XML, JSON, Relational DB	No	No	No	Yes
Web Ontology Building System for Novice Users	RDFS / OWL	Yes – OWL / RDFS	No	No	Yes	N/A
ROO	OWL	No	No	Yes	Yes	N/A
SABer	XQueries, SPARQL queries and Web API calls	Yes	Yes	Yes	Yes	Yes

SABer and SARA work in tandem with each other, and their combined feature sets highlight how they advance the state of the art. As shown in Chapter 4, these features stemmed directly from the design requirements of the framework associated with the *expert-supported approach to data exploration*. This close coupling of design and implementation means that the successful evaluation of SARA and SABer described in this chapter also helps to validate the usefulness of the underlying approach and its framework and models.

5.9 Summary

This chapter described the overall evaluation strategy employed in this thesis, as well as detailing the various experiments involved. These evaluation studies incorporated a variety of techniques, such as user trials, performance tests, questionnaires and interviews, and included experiments with SABer, SARA and the third party applications that used them. An analysis of the evaluation results highlighted how the implementation of SARA and SABer were successful, that these systems fulfilled the research objectives outlined in Section 1.3 and that they contained a set of features that advanced the state of the art. Furthermore, their successful evaluation also helped to validate the *expert-supported approach to data exploration* itself. The following chapter concludes this thesis by outlining the specific contributions that have been made, and by highlighting some future work that can extend this research.

6 Conclusion

This chapter presents the conclusions of the thesis. Initially it examines to what the degree the objectives outlined in this thesis have been fulfilled and then discusses the contributions that this research has made. The work undertaken in this thesis has potential to be extended upon in a number of different areas, thus some possible routes for investigation are also discussed in the Future Work section.

6.1 Objectives & Achievements

The central research question of this thesis examined how *subject matter expertise may be effectively encoded by non-technical experts and then leveraged by casual users to assist exploration and querying of multiple data sources from a domain.* The following objectives stemmed from this research question:

1) Analyse the state of the art in data exploration to determine the extent to which casual users are facilitated, and examine the state of the art in SME encoding for non-technical experts to identify the main features of current approaches.
2) Define an approach that allows end users to leverage SME (tailoring as appropriate) when exploring and consolidating information from separate data sources in a domain, and describe the accompanying models and framework necessary to implement it.
3) Perform evaluation studies to assess:
   a) the usability (effectiveness, efficiency and satisfaction) (Jokela et al. 2003), of the implemented SME authoring component of the framework, and the ability of non-technical users to encode SME.
   b) whether the encoded SME can be usefully exploited by client applications to adequately support end-user exploration.
   c) the features of the framework implementation and whether the consolidation of data from separate sources is supported.

The primary objective of this thesis (Objective 2) was to define an underlying approach and its accompanying framework and models, which allows end users to leverage SME (tailoring as appropriate) when exploring and consolidating information from separate data sources in a domain. This was achieved through the creation of the *expert-supported approach to data exploration*, which underpins a set of models and a novel framework consisting of several components and interfaces. The close coupling of the design outlined in Chapter 3 with its implementation (described in Chapter 4), meant that the successful evaluations of SARA and SABer (detailed in Chapter 5) also validated the usefulness of the approach and its accompanying framework and models.

The design specification outlined in Chapter 3 was influenced by Objective 1 of this thesis, which was to examine the state of the art in how casual users are facilitated in performing data exploration, as well as the state of the art in SME encoding for non-technical experts. From the analysis of deficiencies within the state of art that took place in Chapter 2, the following novel design requirements were derived for the framework which embodies the *expert-supported approach to data exploration*:

- Provide client applications, which support user exploration, with consolidated access to multiple sources of various data types, without prescribing a specific user interaction paradigm for the GUI, or for their developers to know the underlying query language.
- Enable the SME leveraged by the client applications to be encoded by non-technical experts, and to be tailored to an end user's interpretation.
- Enable casual users that appropriate and tailor this SME within the client applications, to send complex queries to multiple data sources via the framework.

Moreover, these requirements were accompanied by these best practice features also derived from the state of the art:

- Allow data to reside in its original location, and provide instance level reconciliation between the different data sources.
- Be a domain independent and modular framework that supports the reusability and movement of sources and SME between different installations.
- Support extensibility by enabling addition of extra SME and of new sources (even of a different data type) to enrich the relationships within the domain.

All of these features were successfully implemented into the framework design described in Chapter 3, and the seven steps that constitute the *expert-supported approach to data exploration* were also detailed within that chapter.

The *expert-supported approach to data exploration* and its accompanying framework and models were used to underpin the technical implementations of the Reconciliation Engine (SARA) and the SME Authoring Tool (SABer). As described in this thesis, SARA was successfully deployed in a number of different domains (Music, Film, Digital Images and Academic Publications), supporting various client applications. Likewise SABer, which supports non-technical domain experts to encode SME, was successfully utilised in a number of domains.

Both SARA and SABer were successfully evaluated (as described in Chapter 5), which fulfilled the final objectives set out for this thesis (3a-3c). Because SARA is essentially a middleware system, its evaluation necessitated its use by a number of client applications. These experiments showed that participants could use these applications to construct complex queries (explicitly and implicitly) over different information sources and to explore the data in a useful fashion (see Sections 5.3 and 5.5.8). Because this functionality was largely facilitated by SARA, it supports the contention that SARA is a useful framework for supporting the querying and exploration of metadata rich domains. Furthermore, a performance evaluation (Section 5.7) of the SARA prototype specified under what conditions the current prototype system is suitable to be deployed. SABer also had evaluation experiments performed on it with end users (see Sections 5.4 and 5.6), and afterwards it was concluded that SABer is a tool with good usability and that domain experts without a computer science or information modelling background could use it to create semantic attributes with a minimal amount of training.

6.2 Contributions to the State of the Art

The major contribution of this thesis is the *expert-supported approach to data exploration* and its accompanying models. This approach contributes to the state of the art by defining a novel and generic knowledge access platform, which serves as a useful intermediary between curators and consumers of data. Specifically this approach supports non-technical experts to define rules relating to notional concepts in their domain, which can then be leveraged and tailored by end users in order to bridge the gap between them and the data they are interested in. These features were highlighted in Chapter 5 where the generation of semantic attributes by non-technical experts, and their successful deployment in various client applications, were described.

By supporting the manipulation of SME by end users, it makes it possible for the continuum between the expert and end user to blur over time. For instance, as the end user becomes more comfortable with the client application and the domain, they can tailor the semantic attributes, allowing the novice user to express their own views of the domain.

Their views may not be "expert" per se, but they may be more appropriate and useful to the user at that specific time than wholly appropriating the encoded expertise of another person. This is important, as it is more flexible than a "*one size fits all*" approach to expertise in domains which may contain widely differing opinions on specific concepts.

The expert rules are encoded in a novel reusable SME model that contributes to the state of the art by natively supporting users who wish to tailor these expert rules to their own interpretation or context, which is often necessary because SME sometimes encompasses rather subjective notions such as "cheap" or "near". Chapter 4 detailed how the design for the Semantic Attribute Model was successfully implemented in XML and could be generated in the SME Authoring Tool called SABer. Furthermore, by appealing to domain experts with limited computer and data modelling skills, the approach proposed in this thesis is attractive to a wide audience of experts in variety of domains, and its use of agile data integration techniques allows reusable dataspaces to be formed. These dataspaces help support the reconciliation of data from multiple sources in a domain, and the use of Domain Superclass Models and Source Models outlined in Chapter 3 facilitate this. The music case study outlined in Section 4.4 also highlighted how these models could be utilised to support a dataspace of five sources, which contains data in three different formats. Overall, the approach presented in this thesis, if widely deployed online, has the potential to help systemise the appropriation of expert judgement by casual users, in order to assist meaningful exploration of a domain.

The minor contribution of this thesis is the framework to support the *expert-supported approach to data exploration*. The implementations of SARA and SABer are instruments of this framework which were used to showcase its features. Chapter 5 detailed several experiments which highlighted how these systems, and the underlying framework, were iteratively designed, and their features were ratified with different stakeholders. Furthermore, both SARA and SABer have been successfully utilised in different domains and with different applications. Specifically they were central technologies used in two M.Sc. theses (Gürel, 2009; Hengster, 2009) and two internship projects within the School of Computer Science and Statistics. SARA and SABer are also in the early stages of being deployed in the Science Foundation Ireland funded AMAS project[52], where they are being used by a multinational publishing house in order to support exploration of their vast

[52] http://kdeg.cs.tcd.ie/amas

e-learning resources (Hampson et al. 2011). One of the aims of this project is to help educators to construct learning modules that are pedagogically consistent with a curriculum, as well as being tailored to the individual needs and preferences of each student. SARA is seen as a key enabling technology for this. Furthermore, SARA and SABer have been identified for deployment into the Science Foundation Ireland funded FAME[53] project, where low level streaming data is to be semantically enriched using SARA, in order to provide key information and alerts to a network management dashboard (Conlan et al. 2010). By supporting the injection of expert knowledge into this system, it will enable ordinary end users to get a better understanding of the data generated within the network, and to have more control over its performance. In summary, the successful deployment and evaluation of SARA and SABer validates the framework they represent, and reinforces the value of the *expert-supported approach to data exploration* and its models.

Evidence of the overall contribution to the state of the art comes in the form of seven peer reviewed publications relating to this thesis:

1. C. Hampson, O. Conlan, Facilitating Casual Users in Exploring Linked Data through Domain Expertise, 22nd International Conference on Database and Expert Systems Applications, DEXA 2011, Toulouse, France, August 2011 [IN PRESS]

   Details the SABer evaluation and the implementation of the Music Case Study

2. C. Hampson, M. Gürel, O. Conlan, Using Expert-Derived Aesthetic Attributes to Help Users in Browsing Image Databases, 22nd International Conference on Database and Expert Systems Applications, DEXA 2011, Toulouse, France, August 2011 [IN PRESS]

   Describes the X2Photo Experiment and how the application interacts with SARA

3. C. Hampson, O. Conlan, V. Wade, Challenges in Locating Content and Services for Adaptive eLearning Courses, Eleventh IEEE International Conference on Advanced Learning Technologies, 2011. ICALT 2011, Athens, USA, 6-8 July 2011 [IN PRESS]

[53] http://www.fame.ie

Discusses SARA and SABer's application to the eLearning domain as proposed in the AMAS project

4. C. Hampson, O. Conlan, Leveraging Domain Expertise to Support Complex, Personalized and Semantically Meaningful Queries Across Separate Data Sources, The IEEE Fourth International Conference on Semantic Computing, Pittsburgh, PA, USA, September 22 - 24, IEEE, 2010, pp305 - 308

   Details the extension of SARA to multiple data types, the evolution of its underlying approach and its application to multiple domains

5. O. Conlan, J. Keeney, C. Hampson, F.P. Williams, Towards Non-expert Users Monitoring Networks and Services through Semantically Enhanced Visualizations, 6th International Conference on Network and Service Management (CNSM 2010), Niagara Falls, Canada, October, 2010, 2010

   Discusses SARA and SABer's application to the Network Management Domain as proposed in the FAME project

6. C. Hampson, O. Conlan, Supporting Personalised Information Exploration through Subjective Expert-created Semantic Attributes, Third IEEE International Conference on Semantic Computing, Berkeley, CA, USA, September 14-16, 2009

   Details the Multi-Source Music Trial and introduces the main features of SARA and its underlying approach

7. C. Hampson, Semantically holistic and personalized views across heterogeneous information sources, Proceedings of the 2nd International Workshop on Semantic Media Adaptation and Personalization, 2nd International Workshop on Semantic Media Adaptation and Personalization, London, 17-18 December, 2007, pp249 - 252

   Describes an early version of the approach which underpins this thesis

These represent the publications so far, but it is intended that further work from this thesis and other collaborative research will be published in appropriate venues in the near future.

6.3 Future Work

This thesis has demonstrated the significant potential that SARA and SABer have in supporting casual users to explore information domains of interest. Hence, the likely direction of this research is to extend the features and performance of both these systems, and have them deployed in interesting domains with real world data. Indeed, as has already been mentioned, SARA and SABer are currently in the early stages of use in the AMAS project which is focused in the eLearning domain, and the FAME project which is based in the network management domain. It is envisaged that further domains and opportunities will be sought to showcase the benefits that SARA and SABer offer in the near future. Eventually, it is planned to release SARA and SARA publicly for any domain experts and application developers to experiment with and utilise. Of equal importance to this software release, is to use these systems to further ratify and extend the *expert-supported approach to data exploration* and its associated framework and models. By ensuring that the approach itself is robust, it will enable a variety of complementing technologies to be built on top of it.

6.3.1 Extending SARA and SABer

Though the current prototype of SARA has been shown to deliver sufficient features and performance for many prospective applications, there are a number of additional components and changes it could benefit from. First of all, support for more data sources of different types, including relational databases and streaming data would significantly add to the number of data sources that SARA could natively support. Whereas adding support for relational databases and SQL to the current prototype of SARA should be relatively straightforward, the addition of support for streaming data in the form of a rule engine like JBoss Rules[54] is likely to be more challenging, and may mean having to revisit the underlying framework and approach. Any support for new types of data sources in SARA will also have to be incorporated into SABer, and the minor user interface suggestions that it received during its evaluation will also need to be fully implemented. The addition of further operators and functions for each of the specific query languages supported by SABer would also provide domain experts with more options while encoding their expert rules. However it will have to be ensured that these additions do not make the tool overly complex for the average user. Finally, SABer should be extended so that it

[54] http://www.jboss.org/drools

supports finished semantic attributes to be combined together into Semantic Attribute Queries that experts can experiment with and label as desired. In effect this will mean that SABer will not only be responsible for generating semantic attributes, but it will also double as a generic client application to test Semantic Attribute Queries against the domain in question.

Another useful component that will be added to the overall framework is an authoring tool for Source Model construction. Currently all source models are encoded in XML by hand, which may result in syntactical or structural errors occurring. Hence, an authoring tool with an easy to use GUI is planned that will automatically generate the XML file from fields inputted by the user, and alert them to any potential errors. As this process is very similar to the XML Semantic Attribute Model generation in SABer, this tool should not be difficult to create. Eventually there may be cause to generate one overall tool that is responsible for creating Source Models and semantic attributes, as well as acting as a generic client application in which Semantic Attribute Queries can be executed against the domain. Such a development would not require any changes to the underlying framework or approach. Finally, individual semantic attributes within a Semantic Attribute Query are currently fired in series within SARA. Likewise, superclass conversion queries are also sent in series. In future, it will be necessary to incorporate multi-threading into the implementation of SARA so that multiple queries can run in parallel, thus decreasing overall query latencies.

6.3.2 Linked Data and Instance Level Reconciliation

One attractive area to explore further is the use of Linked Data with SARA, due to the increasing size and richness of metadata in Linked Data format, and its use of dereferenceable URIs to disambiguate and reconcile instances. Furthermore, Linked Data is currently not being exploited widely by ordinary users due to the lack of user-friendly applications which can access it. Hence, a domain independent system like SARA that supports client applications of different types is a suitable candidate to help bridge the gap between the producers of Linked Data and their many potential consumers. As SARA currently operates by reconciling data at the instance level, this can sometimes limit the reconciliation of data from information sources with different identification schemes. Thus, another worthwhile approach may be to investigate the incorporation of data fusion techniques into SARA to help correlate slightly different identifiers relating to the same instance (e.g. recognizing the id "The Beatles" in one music source as equivalent to "Beatles, The" in another). This would help widen the scope of data sources that can be reconciled with other sources within SARA (support for these sources used on an individual basis, or as part of a Union query with other sources is already implemented). The underlying framework that SARA is based on already supports the use of Linked Data and dereferenceable URIs; however the reconciliation aspect of the framework would need to be extended slightly if data fusion techniques were to be introduced into this process.

6.3.3 Multi-domain Queries

Although this thesis has described applications and scenarios in the context of a single domain, the design of the framework underpinned by the *expert-supported approach to data exploration* (as well as the implementation of SARA) can in fact support multi-domain queries. All that is required is for the superclasses and sources of the different domains to be registered with the same installation of SARA. This will be an interesting avenue to explore in future, as supporting users to correlate data across multiple domains is potentially a very powerful feature. Furthermore, if the use of SARA in different domains becomes more widespread, it may become necessary to create a shared vocabulary of superclasses for each domain to help the reusability of Source Models. Unlike ontologies which have an overall hierarchy and properties associated with each concept, all that would be required is for each superclass to be given a consistent label and associated with a specific domain. No relationships or properties need to be defined for each superclass; hence there is much less scope for disagreement amongst domain experts.

Bibliography

Ambrus, O., Möller, K. & Handschuh, S., 2010. Konduit VQB: a Visual Query Builder for SPARQL on the Social Semantic Desktop. In *Workshop on Visual Interfaces to the Social and Semantic Web (VISSW2010).*

Bangor, A., Kortum, P. & Miller, J., 2009. Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3), pp.114-123.

Berners-Lee, T., 2009. DesignIssues: LinkedData. Available at: http://www.w3.org/DesignIssues/LinkedData.html [Accessed February 21, 2011].

Bizer, C., Heath, T. & Berners-Lee, T., 2009. Linked data-the story so far. *International Journal on Semantic*, 5(3), p.1–22.

Borsje, J. & Embregts, H., 2006. *Graphical Query Composition and Natural Language Processing in an RDF Visualization Interface*. B.Sc. Erasmus University, Rotterdam.

Bozzon, A. et al., 2010. Chapter 14: Building Search Computing Applications. In S. Ceri & M. Brambilla, eds. *Search Computing*. Springer Berlin / Heidelberg, p. 268–290.

Bozzon, A. et al., 2010. Liquid query: multi-domain exploratory search on the web. In *Proceedings of the 19th International Conference on the World Wide Web*. ACM, p. 161–170.

Brabham, D.C., 2008. Crowdsourcing as a model for problem solving. *Convergence: The International Journal of Research into New Media Technologies*, 14(1), p.75.

Braga, D. et al., 2008. NGS: a framework for multi-domain query answering. In *IEEE 24th International Conference on Data Engineering Workshop, 2008. ICDEW 2008.* pp. 254-261.

Braga, D. et al., 2008. Optimization of multi-domain queries on the web. *Proceedings of the VLDB Endowment*, 1(1), pp.562-573.

Brambilla, M. & Ceri, S., 2009. Engineering search computing applications: vision and challenges. In *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*. ACM, p. 365–372.

Brooke, J., 1996. SUS-A quick and dirty usability scale. In P. W. Jordan et al., eds. *Usability Evaluation In Industry*. Taylor & Francis.

Campi, A. et al., 2010. Chapter 9: Service Marts. In S. Ceri & M. Brambilla, eds. *Search Computing*. Springer Berlin / Heidelberg, p. 163–187.

Cimiano, P. et al., 2008. Orakel: A portable natural language interface to knowledge bases. *Data & Knowledge Engineering (DKE)*, 65(2), p.325–354.

Conlan, O. et al., 2010. Towards Non-expert Users Monitoring Networks and Services through Semantically Enhanced Visualizations. In *6th International Conference on Network and Service Management*. Niagara Falls, Canada, pp. 406-409.

Daniel, F., Soi, S. & Casati, F., 2010. Chapter 5: From Mashup Technologies to Universal Integration: Search Computing the Imperative Way. In S. Ceri & M. Brambilla, eds. *Search Computing*. Springer Berlin / Heidelberg, p. 72–93.

Davis, B. et al., 2010. Roundtrip ontology authoring. In A. Sheth et al., eds. *The Semantic Web-ISWC 2008*. Springer Berlin / Heidelberg, p. 50–65.

De Araújo, S.F.C. & Schwabe, D., 2009. Explorator: a tool for exploring RDF data through direct manipulation. In *Proceedings of the Linked Data on the Web Workshop (LDOW2009)*. Madrid.

De Araújo, S.F.C., Schwabe, D. & Barbosa, S.D.J., 2009. Experimenting with Explorator: a Direct Manipulation Generic RDF Browser and Querying Tool. In *Proceedings of Visual Interfaces to the Social and the Semantic Web*. Sanibel Island, Florida.

Denaux, R. et al., 2010. Rabbit to OWL: Ontology Authoring with a CNL-based Tool. In N. Fuchs, ed. *Controlled Natural Language*. Springer Berlin / Heidelberg, p. 246–264.

Ding, Y. et al., 2010. Semantic Web Portal: A Platform for Better Browsing and Visualizing Semantic Data. In A. An et al., eds. *Active Media Technology*. Springer Berlin / Heidelberg, p. 448–460.

Dolbear, C. et al., 2007. The Rabbit Language: description, syntax and conversion to OWL. *Technical Report IRI-0004, Ordnance Survey Research Labs*.

Fischer, T., Bakalov, F. & Nauerz, A., 2009. An Overview of Current Approaches to Mashup Generation. In *Proceedings of the 5th Conference on Professional Knowledge Management*. pp. 254-259.

Goebel, M. & Gruenwald, L., 1999. A survey of data mining and knowledge discovery software tools. *ACM SIGKDD Explorations Newsletter*, 1(1), pp.20-33.

Gürel, M., 2009. *Expert assisted exploration of photographs. Supporting Users in Exploring Visual Media through Subjective Aesthethic Attribtues and Crowd-sourced Tags*. M.Sc.Trinity College Dublin.

Halevy, A., Rajaraman, A. & Ordille, J., 2006. Data integration: The teenage years. In *Proceedings of the 32nd international conference on Very large data bases*. Seoul, Korea: VLDB Endowment, p. 9–16.

Hall, W., 2010. What web science could mean for businesses. *computerweekly.com*. Available at: http://www.computerweekly.com/Articles/2010/04/12/240862/interview-wendy-hall-on-what-web-science-could-mean-for.htm [Accessed February 21, 2011].

Hampson, C., 2007. Semantically Holistic and Personalized Views Across Heterogeneous Information Sources. In *Second International Workshop on Semantic Media Adaptation and Personalization*. London: IEEE, pp. 249-252.

Hampson, C. & Conlan, O., 2011. Facilitating Casual Users in Exploring Linked Data through Domain Expertise. In *22nd International Conference on Database and Expert Systems Applications*. Toulouse, France.

Hampson, C. & Conlan, O., 2010. Leveraging Domain Expertise to Support Complex, Personalized and Semantically Meaningful Queries Across Separate Data Sources. In *IEEE International Conference on Semantic Computing, 2010*. Pittsburgh, USA: IEEE, pp. 305-308.

Hampson, C. & Conlan, O., 2009. Supporting Personalized Information Exploration through Subjective Expert-created Semantic Attributes. In *IEEE International Conference on Semantic Computing, 2009*. Berkeley, USA: IEEE, pp. 384-389.

Hampson, C., Conlan, O. & Wade, V., 2011. (In Press). Challenges in Locating Content and Services for Adaptive eLearning Courses. In *11th IEEE International Conference on Advanced Learning Technologies*. Athens, Georgia, USA: IEEE.

Harth, A. et al., 2007. SWSE: Answers before links. In *Semantic Web Challenge, in conjunction with ISWC / ASWC.*

Hedeler, C. et al., 2010. Chapter 7: Dataspaces. In S. Ceri & M. Brambilla, eds. *Search Computing*. Springer Berlin / Heidelberg, p. 114–134.

Hengster, T.J., 2009. *Implicit and Explicit User Modelling Techniques for Interactive Visual Knowledge Exploration*. M.Sc. Trinity College Dublin.

Hildebrand, M., Ossenbruggen, J.R. & Hardman, L., 2007. An analysis of search-based user interaction on the semantic web. In *Technical Report INS-E0706*. Amsterdam: CWI.

Huynh, D.F. & Karger, D.R., 2009. Parallax and companion: Set-based browsing for the data web. Available at: davidhuynh.net/media/papers/2009/www2009-parallax.pdf [Accessed February 21, 2011].

Huynh, D.F., Miller, R.C. & Karger, D.R., 2008. Potluck: Data mash-up tool for casual users. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4), pp.274-282.

Jokela, T. et al., 2003. The standard of user-centered design and the standard definition of usability: analyzing ISO 13407 against ISO 9241-11. In *Proceedings of the Latin American conference on Human-computer interaction*. pp. 53-60.

Kongdenfha, W. et al., 2009. Rapid development of spreadsheet-based web mashups. In *Proceedings of the 18th international conference on the World Wide Web*. New York, NY, USA: ACM Press, pp. 851-860.

Kovacs, K. et al., 2006. A Methodology for Building Conceptual Domain Ontologies. *Technical Report IRI-0002, Ordnance Survey Research Labs*.

Kozaki, K. & Mizoguchi, R., 2005. A Present State of Ontology Development Tools. *Journal of Japanese Society for Artificial Intelligence*, 20(6), pp.707-714.

Kurgan, L.A. & Musilek, P., 2006. A survey of Knowledge Discovery and Data Mining process models. *The Knowledge Engineering Review*, 21(01), pp.1-24.

Lewis, J. & Sauro, J., 2009. The Factor Structure of the System Usability Scale. In M. Kurosu, ed. *Human Centered Design*. Springer Berlin / Heidelberg, pp. 94-103.

Lopez, V. et al., 2010. Scaling Up Question-Answering to Linked Data. In P. Cimiano & H. Pinto, eds. *Knowledge Engineering and Management by the Masses*. Springer Berlin / Heidelberg, pp. 193-210.

Lopez, V., Pasin, M. & Motta, E., 2005. Aqualog: An ontology-portable question answering system for the semantic web. In A. Gómez-Pérez & J. Euzenat, eds. *The Semantic Web: Research and Applications*. Springer Berlin / Heidelberg, p. 546–562.

Lopez, V. et al., 2009. Cross ontology query answering on the semantic web: an initial evaluation. In *Proceedings of the fifth international conference on Knowledge capture*. ACM, p. 17–24.

Madhavan, J. et al., 2007. Web-scale data integration: You can only afford to pay as you go. In *Proceedings of the Third Biennial Conference on Innovative Data Systems Research*. p. 342–350.

Maedche, A. & Staab, S., 2001. Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*, 16(2), pp.72-79.

Mannila, H. & Gunopulos, D., 2009. ACM TKDD special issue ACM SIGKDD 2007 and ACM SIGKDD 2008. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(4), pp.1-2.

Mei, J. et al., 2008. Making URIs published on Data Web RDF dereferencable. In *7th International Semantic Web Conference*. Karlsruhe, Germany.

Moreno, R., 2004. Decreasing cognitive load for novice students: Effects of explanatory versus corrective feedback in discovery-based multimedia. *Instructional Science*, 32(1), pp.99-113.

Möller, K. et al., 2008. A Visual Interface for Building SPARQL Queries in Konduit. In *International Semantic Web Conference*. Karlsruhe, Germany.

Paiva, S. & Ramos-Cabrer, M., 2010. A new approach to query construction in semantic guided systems. In *Proceedings of the IADIS International Conference on Information Systems*. Porto.

Parramon, J.M., 1989. *Color Theory*, New York: Watson-Guptill Publications.

Polowinski, J., 2009. Widgets for Faceted Browsing. In M. Smith & G. Salvendy, eds. *Human Interface and the Management of Information. Designing Information Environments*. Springer Berlin / Heidelberg, p. 601–610.

Qian, L.J., Zhou, M. & Xu, J.R., 2008. An easy and effective approach to manage radiologic portable document format (PDF) files using iTunes. *AJR. American journal of roentgenology*, 191(1), pp.290-1.

Qu, Y. & Furnas, G.W., 2008. Model-driven formative evaluation of exploratory search: A study under a sensemaking framework. *Information Processing & Management*, 44(2), pp.534-555.

Reason, J., 1990. *Human error*, Cambridge University Press.

Rector, A.L., 2006. Users Are Always Right ... Even When They Are Wrong: Making Knowledge Representation Useful and Usable. In *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning*. Lake District, United Kingdom: AAAI Press, p. 4.

Rouse, W.B., 2003. Need to know-information, knowledge, and decision making. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 32(4), pp.282-292.

Russell, A. et al., 2008. NITELIGHT: A Graphical Tool for Semantic Query Construction. In *Semantic Web User Interaction Workshop (SWUI 2008)*. Florence, Italy.

Saint-Paul, R., Benatallah, B. & Vayssière, J., 2008. Data services in your spreadsheet! In *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*. New York, NY, USA: ACM, pp. 690-694.

Scianta Intelligence, 2005. Data Exploration. Available at: http://scianta.com/technology/data_exploration.htm [Accessed February 21, 2011].

Shen, X., Tan, B. & Zhai, C.X., 2005. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. pp. 824-831.

Taylor, T. & Lubkeman, D., 1989. Applications of knowledge-based programming to power engineering problems. *IEEE Transactions on Power Systems*, 4(1), pp.345-352.

Thompson, C.W., Pazandak, P. & Tennant, H.R., 2005. Talk to Your Semantic Web. *IEEE Internet Computing*, 9(6), pp.75-78.

Vaughan-Nichols, S., 2006. Researchers Make Web Searches More Intelligent. *Computer*, 39(12), p.16–18.

White, R.W., Muresan, G. & Marchionini, G., 2006. Report on ACM SIGIR 2006 workshop on evaluating exploratory search systems. In *ACM SIGIR Forum*. pp. 52-60.

Wong, J. & Hong, J.I., 2007. Making mashups with marmite: towards end-user programming for the web. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA: ACM, pp. 1435-1444.

Yasunaga, S., Nakatsuka, M. & Kuwabara, K., 2010. Web Ontology Building System for Novice Users: A Step-by-Step Approach. In *Second international conference on Intelligent Information and Database Systems*. Hue, Vietnam: Springer-Verlag, p. 134–143.

Zloof, M.M., 1975. Query by example. In *Proceedings of the national computer conference and exposition*. Anaheim, California: ACM, pp. 431-438.

Appendices

Appendix A - Background Technologies

This section briefly describes some of the main technologies associated with the encoding and retrieval of data and knowledge. These technologies are widely used in state of the art systems, and are relevant to the implementation undertaken in this thesis.

A.1 XML and XQuery

Extensible Markup Language (XML)[55] is a set of rules for encoding documents in machine-readable form. It is a flexible way to create common information formats that can be shared on the Internet and elsewhere. XML is "extensible" because, unlike HTML, the markup symbols are unlimited and self-defining. Another key feature of XML is that it allows data to be arranged as nodes in a tree structure that is defined in a schema. This structured hierarchy of marked up data can then be easily queried and manipulated by applications. The usefulness of XML as a format is highlighted by the hundreds of XML-based languages that have been developed in it, including RSS[56], Atom[57], SOAP[58] and XHTML[59].

XQuery[60] is a query and functional programming language that is used to query collections of XML data. It was developed by the XML Query Working Group of the W3C and provides the means to extract and manipulate data from XML documents or any data source that can be viewed as XML (e.g. relational databases). It uses XPath[61] expression syntax to address specific parts of an XML document, and supplements this with an SQL-like FLWOR[62] expression for performing joins.

[55] http://www.w3.org/TR/REC-xml/
[56] http://www.rssboard.org/rss-specification
[57] http://tools.ietf.org/html/rfc5023
[58] http://www.w3.org/TR/soap12-part1/
[59] http://www.w3.org/TR/xhtml11/
[60] http://www.w3.org/TR/xquery/
[61] http://www.w3.org/TR/xpath20/
[62] http://www.w3.org/TR/xquery/#id-flwor-expressions

A.2 RDF/Linked Data and SPARQL

RDF (Resource Description Framework)[63] is a W3C recommendation for creating meta-data structures that define data on the Web. RDF differs from XML in that it is designed to represent knowledge, not data, and thus is much more concerned with meaning. RDF is also designed to work with distributed data and achieves this by linking documents together by the common vocabularies they use, and allowing facts to be described by information drawn from multiple sources. An RDF document is composed of triples: (1) the subject (what the data is about), (2) the predicate (an attribute of the subject) and (3) the object (the actual value). A collection of RDF statements thus represents a labelled, directed multi-graph. As such, an RDF-based data model is more naturally suited to certain kinds of knowledge representation than the relational model or other ontological models. However in practice, RDF data is often persisted in relational database or in native representations such as triplestores.

Linked Data (Bizer et al. 2009) is a term used to describe a recommended best practice for exposing, sharing, and connecting pieces of data, information, and knowledge on the Semantic Web using Uniform Resource Indicators (used to identify Internet resources) and RDF. Tim Berners-Lee outlined four principles of Linked Data in his design issues discussion on Linked Data (Berners-Lee 2009), which can be paraphrased along the following lines:

1. use URI Indicators to identify things.
2. use HTTP URIs so that these things can be referred to and looked up ("dereferenced") by people and user agents.
3. provide useful information about the thing when its URI is dereferenced, using standard formats such as RDF/XML.
4. include links to other, related URIs in the exposed data to improve discovery of other related information on the Web.

The goal of the W3C Linking Open Data community project[64] is to extend the Web with a data commons by publishing various open datasets as RDF on the Web and by setting RDF

[63] http://www.w3.org/RDF/
[64] http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData

links between data items from different data sources. By September 2010 this had grown to over 19.5 billion RDF triples[65].

SPARQL (SPARQL Protocol and RDF Query Language)[66] is a protocol and query language for RDF that became an official W3C Recommendation in 2008. It can be used to express queries across different data repositories, whether the data is stared as natively as RDF or viewed as RDF via a conversion wrapper. SPARQL allows for queries to consist of triple patterns, conjunctions, disjunctions, and optional patterns and is seen as a key Semantic Web technology. SPARQL also facilitates users to write globally unambiguous queries through the use of dereferenceable URIs, and federated queries that can be distributed against multiple SPARQL endpoints. Results from SPARQL queries can be expressed as result sets or RDF graphs.

A.3 OWL

OWL (Web Ontology Language)[67] is designed for use by applications that need to actually process the content of information instead of simply presenting the information to users. OWL facilitates greater machine interpretability of Web content than that supported by XML and RDF by providing additional vocabulary along with a formal set of semantics. It has three increasingly-expressive sub-languages: OWL Lite, OWL DL, and OWL Full which have a corresponding increase in language complexity. The SPARQL query language can be used to access data stored in any of the OWL formats. OWL is typically used to develop domain ontologies. These ontologies consist of a set of classes and a set of property assertions, which relate these classes to each other. Such ontologies also have a set of axioms which place constraints on sets of classes and the types of relationships permitted between them. These axioms provide semantics by allowing systems to infer additional information based on the data explicitly provided. This support for inference is what makes OWL such a popular format to encode domain knowledge.

[65] http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/DataSets/Statistics
[66] http://www.w3.org/TR/rdf-sparql-query/
[67] http://www.w3.org/2004/OWL/

Appendix B – XML Representation of Two Songs from a Transformed iTunes Music Library

```
<AudioTrack>
  <TrackTitle>Just</TrackTitle>
  <MusicArtistName>Radiohead</MusicArtistName>
  <AlbumArtist>Radiohead</AlbumArtist>
  <Composer>Colin Greenwood/Ed O'Brien/Jonny Greenwood/Phil
            Selway/Thom Yorke
  </Composer>
  <AlbumTitle>The Bends</AlbumTitle>
  <Genre>Rock</Genre>
  <AudioFileType>MPEG audio file</AudioFileType>
  <FileType>MPEG audio file</FileType>
  <AudioFileSize>3738939</AudioFileSize>
  <TrackDuration>233508</TrackDuration>
  <TrackNumber>7</TrackNumber>
  <YearOfTrack>1995</YearOfTrack>
  <TrackBitRate>128</TrackBitRate>
  <TrackSampleRate>44100</TrackSampleRate>
</AudioTrack>
<AudioTrack>
  <TrackTitle>Life Becoming a Landslide</TrackTitle>
  <MusicArtistName>Manic Street Preachers</MusicArtistName>
  <AlbumArtist>Manic Street Preachers</AlbumArtist>
  <Composer>James Dean Bradfield/Nicky Wire/Richey James/Sean
            Moore
  </Composer>
  <AlbumTitle>Gold Against the Soul</AlbumTitle>
  <Genre>Rock</Genre>
  <AudioFileType>MPEG audio file</AudioFileType>
  <FileType>MPEG audio file</FileType>
  <AudioFileSize>4082166</AudioFileSize>
  <TrackDuration>254955</TrackDuration>
  <TrackNumber>5</TrackNumber>
  <YearOfTrack>1993</YearOfTrack>
  <TrackBitRate>128</TrackBitRate>
  <TrackSampleRate>44100</TrackSampleRate>
</AudioTrack>
```

Appendix C – XML Representation of Two Songs from US Singles Charts (1950-2008)

```
<ChartingSong>
  <Year_Released>1964</Year_Released>
  <Yearly_Rank>79</Yearly_Rank>
  <Num_Weeks_Charted>13</Num_Weeks_Charted>
  <Num_Weeks_Charted_Top40>10</Num_Weeks_Charted_Top40>
  <Num_Weeks_Charted_Top10>3</Num_Weeks_Charted_Top10>
  <Num_Weeks_At_Its_Peak_Place>1</Num_Weeks_At_Its_Peak_Place>
  <Highest_Charting_Position>7</Highest_Charting_Position>
  <Artist_Name>Johnny Tillotson</Artist_Name>
  <Track_Title>Talk Back Trembling Lips</Track_Title>
  <Label>MGM 13181</Label>
  <Genre>Rock</Genre>
  <BeatsPerMinute>137</BeatsPerMinute>
  <Written_By>John D. Loudermilk</Written_By>
  <Date_Entered_Charts>23265</Date_Entered_Charts>
  <Date_Peaked_Charts>23468</Date_Peaked_Charts>
</ChartingSong>
<ChartingSong>
  <Year_Released>1964</Year_Released>
  <Yearly_Rank>80</Yearly_Rank>
  <Num_Weeks_Charted>14</Num_Weeks_Charted>
  <Num_Weeks_Charted_Top40>11</Num_Weeks_Charted_Top40>
  <Num_Weeks_Charted_Top10>5</Num_Weeks_Charted_Top10>
  <Num_Weeks_At_Its_Peak_Place>2</Num_Weeks_At_Its_Peak_Place>
  <Highest_Charting_Position>8</Highest_Charting_Position>
  <Artist_Name>Jan and Dean</Artist_Name>
  <Track_Title>Dead Man's Curve</Track_Title>
  <Label>Liberty 55672</Label>
  <Genre>Rock</Genre>
  <BeatsPerMinute>137</BeatsPerMinute>
  <Written_By>Jan Berry, Roger Christian, Artie Kornfeld, Brian
            Wilson
  </Written_By>
  <Date_Entered_Charts>23561</Date_Entered_Charts>
  <Date_Peaked_Charts>23625</Date_Peaked_Charts>
</ChartingSong>
```

Appendix D – Multi-Source Music Trial Questionnaire

Section A

On a scale of 1-10 (with **1 equalling total disagreement** and **10 equalling absolute agreement**) How much do you agree with the following statements?

1. *Using this framework is a more efficient way of finding information than having to consult the individual data sources separately?*

   ____

2. *Using this framework it is easy to combine data from different sources?*

   ____

3. *Using this framework allows users to combine and interchange attributes (popularity/duration/freshness etc) easily?*

   ____

4. *Using this framework enables knowledge to be gained that wouldn't be possible by consulting just a single source?*

   ____

5. *This framework sufficiently enable users to personalise and tailor their queries (a very popular song is top 10, a long song is more than 5mins, etc)*

   ____

6. *The approach used by the framework is very applicable to other domains beside music?*

   ____

   - Do you have any suggestions for different domains?

     ____

     ____

SECTION B

1. If you were to develop an application that used an API into this framework, what features would you like to see incorporated in it?

2. How beneficial would a composition tool be that helped developers combine data sources and define semantic attributes for different domains? What features should it have?

3. Do you have any other suggestions or comments (features you liked/disliked, missing aspects you would like to see)?

Appendix E – Interview for Multi-domain Evaluation of SABer and SARA

1. What features of SABer did you like?
2. What features of SABer did you dislike?
3. What features of SABer would you like to see?
4. What features of the API did you like?
5. What features of the API did you dislike?
6. What features of the API would you like to see?
7. What are the main benefits of using SARA to mediat between your application and data sources?
8. Are there any disadvantages to using SARA as a mediator between your application and data sources?
9. Do you have any other suggestions of comments you would like to make?

Appendix F - SABer Evaluation Questionnaire

1. How did you find the process of creating Semantic Attributes.

Very Difficult | | | | Very Easy

1	**2**	**3**	**4**	**5**

Comment:

____

2. Was it considerably more difficult to make rules of Query type A, B or C.

Yes | | | | No

1	**2**	**3**	**4**	**5**

If so please state which one(s)

____

3. Was it considerably more difficult to create semantic attributes of type *Expert*, *Hybrid* or *Template*.

Yes | | | | No

1	**2**	**3**	**4**	**5**

If so please state which one(s)

____

4. Do you think, with minimal training, that it is likely that an “expert” with no computer coding experience could use this tool to create semantic attributes in their domain.

Very Likely | | | | Very Unlikely

1	**2**	**3**	**4**	**5**

Comment:

____

5. Do you have any suggestions regarding the usability of the authoring tool and its interface?

6. Do you have any suggestions regarding functionality you would like to see?

Appendix G - System Usability Scale

	Strongly disagree				Strongly agree
1. I think that I would like to use this system frequently	1	2	3	4	5
2. I found the system unnecessarily complex	1	2	3	4	5
3. I thought the system was easy to use	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this system	1	2	3	4	5
5. I found the various functions in this system were well integrated	1	2	3	4	5
6. I thought there was too much inconsistency in this system	1	2	3	4	5
7. I would imagine that most people would learn to use this system very quickly	1	2	3	4	5
8. I found the system very cumbersome to use	1	2	3	4	5
9. I felt very confident using the system	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this system	1	2	3	4	5

Appendix H – X2Photo User Experiment

Experimental Setup

A user trial was devised to help elicit the experiences of people who used X2Photo. With this aim in mind the following approach was pursued: four photographs (see Figure I-1) not present in the 12,000 photograph collection connected to X2Photo were selected, and then annotated by the domain expert with different semantic attributes from Table 5-3. The nine participants in the user trial were each shown these four photographs initially and asked to describe them in their own words. Then they were given an overview of X2Photo and asked to do the following tasks:

- For photograph 1, find similar photographs via X2Photo
- For each photograph found, add it to the Favourites.
- Repeat this task for all four photographs.
- Once complete, go to Flickr and for photograph 1 try to find similar images either with the words originally used to describe the photographs or with different ones.
- Repeat these tasks for all four photographs.

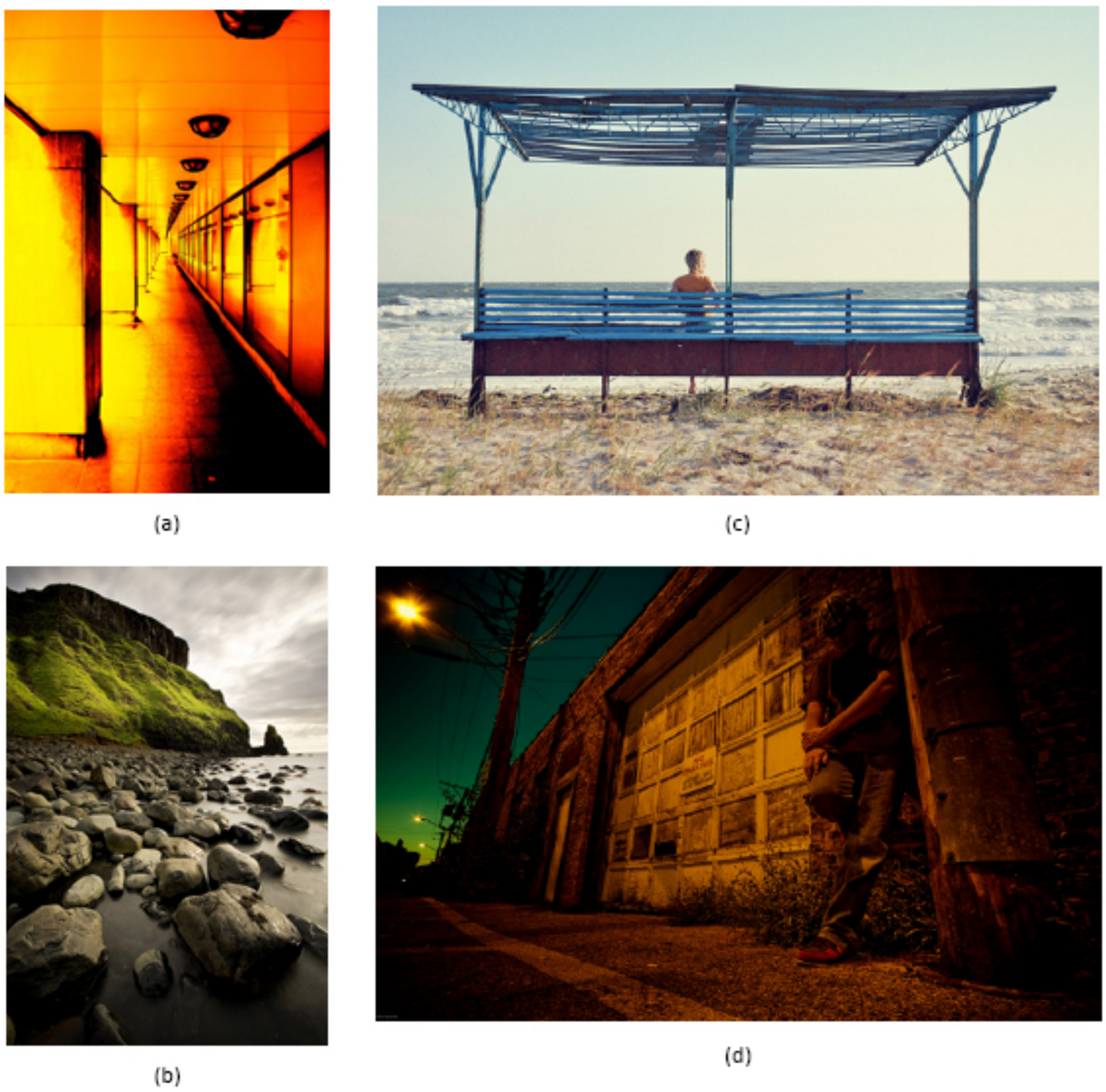


Figure I-1 Four initial photographs shown to users

Experimental Results

Once the given tasks were completed, a user survey was conducted (see Appendix J). The questionnaire intended to evaluate each feature's functionality and the overall system quality in various aspects. The general response to the usability and appeal of the Discovery Space was very positive, agreeing that the continuous flow enabled them to browse the photographs thoroughly, and that the interaction with the space was appealing. Below is a summary of the other survey results:

- 8/9 users considered the overall UI to be "very good" when asked on a four point Likert scale (Very Good, Good, Poor, Very Poor).
- 8/9 "strongly agreed" that the system was attractive when asked on a four point Likert scale (Strongly Agree, Agree, Disagree, Strongly Disagree).

- 8/9 users found the zoom effect in the interface to be “very good” or “good” when asked on a four point Likert scale (Very Good, Good, Poor, Very Poor).
- 9/9 users thought the AttBar was “very good” or “good” when asked on a four point Likert scale (Very Good, Good, Poor, Very Poor). It was also mentioned that the concept off the AttBar was comprehensible and that the classification of the attributes was clear.
- 8/9 users found the ability to refine a search with a focus image to be very useful or useful when asked on a four point Likert scale (Very Useful, Useful, Not Useful, Completely Useless). Seeing what semantic attributes and tags were associated with the focus image allowed them to use these as a springboard for their browsing.

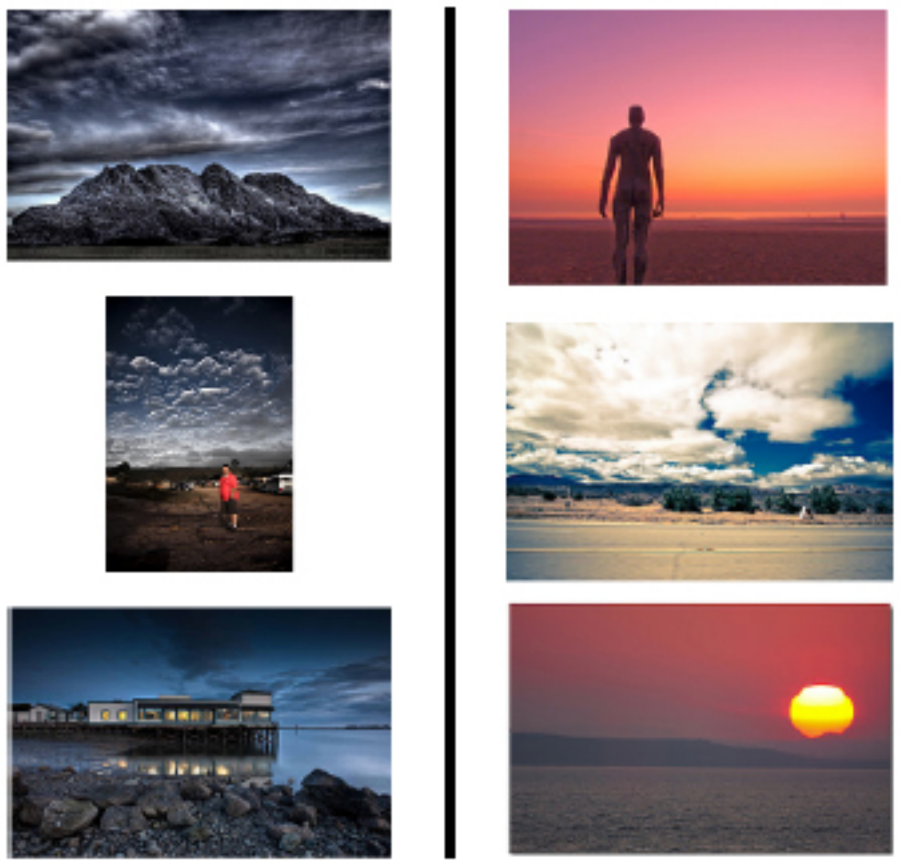
Describing the Images

The majority of users who evaluated X2Photo were technically proficient with computers and four considered themselves to be amateur photographers. The way in which users described the four photographs had some noteworthy aspects, for example those users who were interested in photography tended to use more technical phrases. These included terms such as “over-exposed” when describing photograph *a*, mentioning the angle at which photograph *b* might have been shot, and questioning whether photograph *b* was altered in an image editing program to obtain its deep contrast. These users tended not to describe the content of the photograph as much as the users with less photography experience. Some users preferred to describe the photographs with more personal expressions such as “lonely” and “tempting” when referring to photograph *c*. Photograph *d*, as expected, was interpreted differently by almost all the users. While some tried to identify what the man in the picture might be doing, some chose to describe his character/mood, resulting in many different impressions such as “gritty”, “relaxed” or “run-down”.

Almost all the users first chose expressions like “warm”, “cold”, “airy”, “gloomy” and “energetic”, some of which directly coincided with the actual attributes determined by the domain expert. They then proceeded to describe the actual content. Two of the nine participants were more objective in their descriptions and chose to name the elements they saw in the photographs with words like “corridor”, “bench”, “rocks”, “back alley”, etc. However, the vast majority of users combined their perceptions with the content: “...a cool calm picture but alive… there's a woman sitting on a bench... feels breezy but soft... waves look relaxing”.

Finding Images in X2Photo

In the same way that users' manner of describing photographs differed, so to did their approach to finding similar photographs in X2Photo. Four users never actually used the TagBall which contained all the associated tags relating to the photos from Flickr. Coincidentally their descriptions of the photographs were heavily reliant on expressions like "moody", "dark", "calm", etc. They directly chose similar words present within the AttBar (the component of the GUI which listed the semantic attributes from Table 5-3) and then carried out their searches. After receiving their initial results two of these users were surprised to see how the tool interpreted their descriptions. They did not agree with the expert and started experimenting with the AttBar rather than continuing with their searches. After observing some consecutive result sets and bringing some photographs into focus (which displayed the semantic attributes and tags associated with the image), they stated that they grasped the association the expert was making and modified their searches accordingly. The other two users who didn't use the tag ball performed 2-3 consecutive searches, which were refined each time, to find a similar photograph. On observation of the similar photographs that users selected, it was interesting to see what the users based their similarity criteria on. While some photographs had a similar feel to them regarding the concept or the context, some were also similar in content. Figure I-2 shows examples of similar pictures (of images b and c in Figure I-1) found by users using the TagBall and AttBar in X2Photo



Similar to photograph (b) Similar to photograph (c)

Figure I-2 Examples of similar pictures found by users using X2Photo

Finding Images in Flickr

When the users tried to find similar photographs in Flickr, their approaches were again different. For example, one user used "fiery clinical harsh" to search for similar photographs to photograph *a*, which was the expression he had used when describing the photograph originally. In contrast, another abandoned the expressive vocabulary he used originally to describe photograph *b* (because he was very familiar with searching on Flickr) and chose to use the search phrase "Scotland cliff coast". Changes in vocabulary were noticed with three user's searches within Flickr. For instance, a user who had previously described photograph *c* mainly based on its content: "beach, person sitting on the bench, greyish" found a similar image within X2Photo that the expert thought to be "romantic", "soothing" and "innocent". Using Flickr, the user changed his first search to be "romantic sea scenery". Users familiar with Flickr also used the advanced search available and refined their queries, but again tended to use content-based terms to carry out their searches. In the end all the users were able to find at least one similar image for each of the photos, which was not surprising considering the sheer volume of photographs in Flickr.

However what was noteworthy was the fact that users had to resort to content-based terms which were identical in many cases. This implies that the users were more limited in their expressivity when searching.

Analysis

The user test suggested that when describing photographs people like to communicate subjective descriptors as well as the subject matter of the photo. This finding indicates a need for a wider vocabulary to be available to users in order to retrieve accurate and relevant photographs from any collection. Traditional tag-based systems tend to be dominated by content-based terms, and ignore the artistic quality of a photo which can be a key factor in evoking appreciative emotions. These systems often reduce photographs to a list of mainly content-based words. Since most people have become accustomed to this approach, they tend to ignore other ways in which they could approach a photograph, and are therefore reduced to searching for tagged simplifications of photographs, rather than the photographs themselves.

Appendix I – Film Domain Exploration Client User Experiment

Experimental Setup

After a brief introduction, seven participants were asked to interact with the application. Five of these were male, and two were female. All of them were of age 24 to 30 and all were computer literate. The only task they were assigned was to "find 25 to 30 films you like through exploring". Afterwards a detailed questionnaire was filled in by all users. A trial session took approximately 40 - 55 minutes. This included the introduction, using the application, and filling out the questionnaire (see Appendix K). The interaction with the tool took on average 20 minutes.

Experimental Results

Interestingly, none of the users kept direct track of how many films they had collected as they went along, and all but one of the users had to be prompted to finish interacting with the tool, as users were collecting many more films than were necessary for this evaluation. This supports the notion that they were involved in a continuous exploration of the poster images and were enjoying the experience. The following quotes from the user questionnaire would support this:

- "I like the exploration!"
- "Could play with it for hours!"
- "System returned movies I haven't seen in a long time."
- "I like the variety of poster images shown."
- "Easy to build catalogue of liked/disliked films. IMDb (Internet Movie Database) does not do this."
- "I'm still curious about the reasoning behind it!"
- "Nice application! Could use it!"
- "Cool idea!"
- "Put into IMDb now!"

Though the user enjoyed their use of the application there were a number of suggestions as to how to improve the interface such as:

- Allowing certain films to be ignored.

- Having a search bar which allows a new focus film to be inputted.
- Adjusting the rating system to a more intuitive plus/minus metaphor rather than a push/pull one.
- Allowing users to navigate using the keyboard arrows.
- Having a button to quickly browse the list of liked and disliked films.

However, all volunteers agreed that it was simple to rate films and agreed that the interface was fun to use. One participant commented on the “spiral effect” which they really liked, because it helped identifying the focus film and the relations on screen. All participants strongly agreed to have found films they liked. Furthermore, nearly everyone agreed to have found films they expected. The two users who were “surprised” by most of the films presented to them expressed that this was “not a bad thing” as they liked to see films they weren’t expecting. Thus in general, people were happy with the films that had been shown to them. Moreover, the majority emphasised that they had been shown films they hadn’t thought about for a long time and would have wanted to watch some of them if they had the time.

At the end of the questionnaire, users were explicitly asked what they thought the application was good for, or what it could be used for:

- “Film exploration and getting a map of everything you have seen”
- “Suggesting a movie and then finding a movie to watch”,
- “Finding recipes! The system is based on some logic which is bringing up unexpected results. It’s like to be surprised continuously [*sic*]!”
- “Wish list!”
- “Finding new films to watch, especially ones you haven’t heard of and haven’t seen!”
- “Ability to find resembling movies [*sic*] that either you haven’t seen or you missed!”
- “Good to remember films that you didn’t think of at the moment. It can be used as a favourites list!”
- “Link films to external sources like IMDb or rental stores would make it easy to select films you may like”
- “Add additional information to films to assist with decision making”
- “Add comments to any movie” was mentioned a few times”

- “Browse movies by genre and other criteria” was high on the wish list

The results that were gathered exhibited two general characteristics; apart from user interface and interaction issues which created some confusion, all volunteers were able to gain sufficient immersion and enjoyed using the tool to explore films. There were some suggestions for improvement, but it can be said that all participants liked the user interface and navigation in general.

Appendix J – X2Photo User Questionnaire

1. Do you consider yourself a photography enthusiast?
2. Do you have an account in an online photo sharing site such as Flickr, Picasa, etc.?
3. If so do you tag your photos and with what type of associations?
4. Do you have a personal blog in which you display photographs?
5. If so what kind of methods do you use to annotate them? (Tag them, use captures, titles, etc.)
6. When you need to find images, which image search engines or stock photography sites, or any other, do you use?

Please rate the following:

Discovery Space

	Very Good	Good	Poor	Very Poor
TagBall				
AttBar				
Zoom				
Extra Details				
Focus				
Favorites Area				
Overall UI				

During exploration I found:

	Very Useful	Useful	Not Useful	Completely Useless
Using Tags				
Using Atts				
Refining a Search based on focus image				

Overall Search

I found the system to be:

	Strongly Agree	Agree	Disagree	Strongly Disagree
Attractive				
Powerful				
Empowering				
Frustrating				
Responsive				
Slow				
Extensive				
Confusing				
Straightforward				

Comments:

Appendix K - Film Domain Exploration Client User Questionnaire

Personal

1.	Name	
2.	Age	
3.	Gender	Male Female
4.	How comfortable are you with computers?	Not at all ¨ A little ¨ Somewhat ¨ Moderately ¨ Quite a lot ¨ Very much ¨

Films in general

5.	I am interested in films.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
6.	I know a lot about films.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
7.	I watch more films now compared to 10 years ago.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
8.	I watch a lot of films.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
9.	I regularly get information about films from various sources.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
10.	Where do you usually hear about new films?	On the radio ¨ On TV ¨ In the cinema ¨ Online ¨ In magazines/ newspapers ¨ Don't Know ¨ Other ………………………………
11.	How often do you use online services or websites to get information about films?	Very often ¨ Often ¨ Rarely ¨ Very rarely ¨ Not at all ¨

12.	Please select any of the following film related websites you have **heard of**!	IMDb ¨ Freebase ¨ Rotten Tomatoes ¨ MovieLens ¨ What To Rent ¨ Criticker ¨ Clerkdogs ¨ Jinni ¨ Other ....
13.	Please select any of the following film related websites you have **used at least once**!	IMDb ¨ Freebase ¨ Rotten Tomatoes ¨ MovieLens ¨ What To Rent ¨ Criticker ¨ Clerkdogs ¨ Jinni ¨ Other ....
14.	Please select any of the following film related websites you **use regularly!**	IMDb ¨ Freebase ¨ Rotten Tomatoes ¨ MovieLens ¨ What To Rent ¨ Criticker ¨ Clerkdogs ¨ Jinni ¨ Other ....
15.	When browsing for films, what types of information are you interested in most?	General information New films Upcoming films ¨ Actors ¨ Popular films ¨ Films from specific genre ¨ Don't know ¨ Other ....
16.	I like "Blockbuster films".	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
17.	I like "Expensive films".	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
18.	I like "Award winning films".	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagre ¨
19.	I like "Successful films".	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
20.	I like...	

Software client / UI

21.	It is easy to select an initial film with the tool I just used.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨

22.	It is clear that I can change the initial film before I continue.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
23.	It is easy to rate films.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
24.	Setting a new focus film is simple to do.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
25.	It is intuitive to zoom.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
26.	Panning (navigating the stage vertically and horizontally) is easy.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
27.	It is easy to get lost.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
28.	Showing the film poster image is useful.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
29.	Showing the film title on the poster helps indentifying the films.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
30.	It is simple to discover films I previously liked/disliked.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
31.	The user interface makes it easy to explore films.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
32.	The user interface is fun to use.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨

Further questions / general statements

33.	It was difficult to come up with an initial film in the first step when using the program.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
34.	I found films I like.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
35.	I found films I expected the software will show to me.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
36.	I think the system worked well.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
37.	The system was predictable / it is clear what is going on *"under the hood"*.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
38.	Do you remember how often you refocused on a new film?	Yes ¨ Number: .... No ¨

39.	Exploring films this way is a waste of time.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
40.	I am happy with the films that have been shown to me.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
41.	I wasn't aware that I know that many films.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
42.	I was shown films I haven't been thinking about for a long time.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
43.	If I had time right now I would like to watch one of the films I found.	I strongly agree ¨ I agree ¨ I disagree ¨ I strongly disagree ¨
44.	What do you think the system is good for? What can it be used for?	
45.	What would you change? (Navigation, Rating, Visualisation...)	
46.	What do you think are the most significant remaining usability issues?	
47.	What do you like most/least about the film exploring tool?	
48.	How could the system be improved overall? Other features?	
49.	Any other comments, ideas, questions?	