

Exploring Query Augmentation for the SOCRADES Application Service Catalogue

Master thesis in Information Management

MASTER THESIS

BETTINA DOBER

March 2009

Thesis supervisors:

Dominique GUINARD
SAP Research Zurich

Prof. Dr. Jacques PASQUIER & Dr. Patrik FUHRER
Software Engineering Group
Department of Informatics – University of Fribourg (CH)

Abstract

Production facilities still have to become more efficient and more effective. The automation and total integration of software in the operations management is necessary to be able to fully accommodate the customers' demand for flexibility. The trend towards the Internet of Things contains potential for solving that problem by integrating Smart Objects over strategic, tactical and operational business level. Smart Objects in this context is a general term for a web service enabled devices, sensor networks, RFID devices and their applications.

The EU research project SOCRADES is pursuing exactly these goals, so that sophisticated production and monitoring processes can be modelled with it. SOCRADES proposes an entire service oriented production system consisting of Smart Objects integrated into the businesses software. The SOCRADES integration architecture integrates Smart Objects via web services.

We do not treat the whole SOCRADES integration architecture in this master thesis but we present one central question of the project: How can one find the right service that meets the specific needs of the current enterprise application or current user? – Web service directories following the UDDI standard do not offer advanced search functionalities for the end users. Therefore we propose a new query method based on a number of collaborating components in order to provide a way of searching for functionalities on devices. The main component involved is the Application Service Catalogue, which gives assistance to the users' search. It consists of several subcomponents, amongst others the Query Augmentation Module. The query augmentation is the first step of the search for a web service and consists, in a nutshell, of enriching a simple user query.

The main topic of this master thesis is the development of this Query Augmentation Module and in particular the definition of enrichment strategies, the creation of the component architecture, the first implementation and its evaluation. With strategies we mean methods of query expansion. The strategies of the Query Augmentation Module are consuming available web resources, which are analyzed, evaluated and then used for the query expansion. Possible bases for the query expansion are analysed Wikipedia articles or evaluated search results from common search engines. The strategies that are discussed in this report are kept simple, but the Query Augmentation Module consists of a very flexible design and more sophisticated strategies can be added easily at any time.

Furthermore, we present the benefit of using the Query Augmentation Module within the SOCRADES integration architecture.

Keywords: Query Augmentation, Query Expansion, Wikipedia, Web Services, SOCRADES

Preamble

Foreword

The master thesis “Exploring Query Augmentation for the SOCRADES Application Service Catalogue” was performed by Bettina DOBER during the period from May to November 2008 as part of an internship at SAP Research CEC Zurich. Therefore this master thesis results from a cooperation of the University of Fribourg and the SAP Research CEC Zurich. At University the Software Engineering Group was supervising this master thesis.

The work consists of the design, implementation, and evaluation of the Query Augmentation module, which is a subcomponent of the European research project SOCRADES [1].

Acknowledgement

Thanks are due to all those who accompanied me during the entire study and in the time of my master thesis.

In particular I would like to thank Prof. Dr. Jacques PASQUIER and Dr. Patrik FUHRER at the University of Fribourg as well as Dominique GUINARD at SAP Research CEC Zürich for the opportunity of realizing my master thesis in the Software Engineering Group in collaboration with SAP Research. Dominique GUINARD, my supervisor at SAP Research, and Dr. Patrik FUHRER, my supervisor at University, helped me with their professional advises, suggestions, precious reviews. Thanks a lot for all that very valuable support.

A special thank goes to Patrik SPIESS and the whole team of the SOCRADES project for the active support during the implementation phase of my master thesis.

Finally I want to thank the whole SAP Research team for providing me an insight into the industrial research.

About this documentation

This documentation is based on a template created by Patrik FUHRER and Pedro DE ALMEIDA (Software Engineering Group, University of Fribourg, Switzerland).

The template is open source and freely available from [2].

Table of Contents

| | |
|---|-----------|
| 1 Introduction..... | 1 |
| 1.1 SOCRADES | 2 |
| 1.2 Application Service Catalogue | 4 |
| 1.3 Query Engine | 5 |
| 1.3.1 Why query augmentation? | 5 |
| 1.3.2 Query Augmentation Module..... | 6 |
| 1.4 Research questions..... | 6 |
| 1.5 Formal structure..... | 7 |
| 1.6 Conventions | 7 |
| 2 Related Work..... | 9 |
| 2.1 Query Expansion | 9 |
| 2.2 Consuming Wikipedia | 10 |
| 2.3 Semantic Web..... | 12 |
| 3 Architecture | 14 |
| 3.1 Query strategies | 15 |
| 3.1.1 Wikipedia Strategy | 17 |
| 3.1.2 Yahoo Strategy | 17 |
| 3.1.3 DBpedia Strategy | 19 |
| 3.2 Query Augmentation Module | 19 |
| 3.2.1 Strategy design pattern | 20 |
| 3.2.2 Template method design pattern | 21 |
| 3.2.3 Dynamic class loading | 23 |
| 3.2.4 Architectural overview of the Query Augmentation..... | 25 |
| 3.3 Query Augmentation GUI | 26 |
| 3.4 Future Work..... | 26 |
| 3.4.1 Tagging..... | 26 |
| 4 Implementation..... | 27 |
| 4.1 SAP NetWeaver..... | 27 |
| 4.1.1 SAP NetWeaver Application Server | 28 |
| 4.1.2 SAP NetWeaver Development Studio | 29 |
| 4.1.3 Competitive products | 30 |
| 4.2 Java EE | 31 |
| 4.2.1 Java EE application model | 31 |
| 4.2.2 Enterprise JavaBeans..... | 32 |
| 4.2.3 Java Persistence API | 33 |
| 4.2.4 JavaServer Faces | 35 |

| | | |
|----------|---|-----------|
| 4.3 | RESTful Web Services | 39 |
| 4.3.1 | Yahoo! Web Services | 39 |
| 4.3.2 | MediaWiki API | 40 |
| 5 | User manual | 41 |
| 5.1 | Search Service Instances | 42 |
| 5.2 | User Preferences | 43 |
| 5.3 | Stop Words | 43 |
| 5.4 | Configuration | 44 |
| 5.5 | Query Augmentation Summary | 47 |
| 5.6 | Screencasts | 48 |
| 6 | Experimental Evaluation | 49 |
| 6.1 | Methodology | 50 |
| 6.1.1 | Data collection | 50 |
| 6.1.2 | Test data | 52 |
| 6.1.3 | Scenarios | 53 |
| 6.2 | Results | 53 |
| 6.2.1 | Best configuration | 54 |
| 6.2.2 | Wikipedia strategy | 57 |
| 6.2.3 | Yahoo Related Tags strategy | 59 |
| 6.3 | Discussion | 60 |
| 6.3.1 | Future evaluation | 61 |
| 7 | Conclusion | 62 |
| 7.1 | Outlook | 62 |
| 7.2 | Final Thoughts about Query Augmentation | 62 |
| 7.3 | Realisation of the master thesis | 63 |
| A | Project schedule | 65 |
| B | Smart Devices | 66 |
| C | Questionnaire for User Study | 69 |
| D | WSDL File | 74 |
| E | Deployment with NWDS | 79 |
| | Preconditions | 79 |
| | NWDS workspace | 79 |
| | SAP NetWeaver Application Server | 80 |
| | NWDS preferences | 80 |
| | Deployment process | 81 |
| | Add and Remove Projects | 81 |
| | Publish projects | 82 |
| F | Common Acronyms | 83 |
| G | CD-ROM | 85 |
| | References | 86 |
| | Declaration | 90 |

List of Figures

Figure 1.1 The SOCRADES integration architecture (taken from [1]) 2

Figure 1.2 SOCRADES middleware (cf. [1]) 3

Figure 1.3 Application Service Catalogue overview [1]..... 4

Figure 1.4 Query Engine 5

Figure 3.1 Architectural overview in style of Figure 1-1 of [28]..... 15

Figure 3.2 Strategy design pattern [5] 20

Figure 3.3 Template Method design pattern [5]..... 21

Figure 3.4 The hook method doGetSummary 23

Figure 3.5 Architectural overview of the Query Augmentation 25

Figure 4.1 Architecture in the style of figure 1-1 of [28]..... 32

Figure 4.2 JSF datatable 38

Figure 5.1 UI - Menu..... 42

Figure 5.2 UI - Search for Service Instances 42

Figure 5.3 UI - Stop Words 44

Figure 5.4 UI - Configuration 45

Figure 5.5 UI - Query Augmentation Summary – part I..... 47

Figure 5.6 UI - Query Augmentation Summary – part II..... 48

Figure 6.1 Trend of Query Augmentation..... 55

Figure 6.2 Augmentation strategies 57

Figure 6.3 Existence of Wikipedia Articles 58

List of Tables

| | |
|--|----|
| Table 3.1 Wikipedia strategies | 17 |
| Table 3.2 Strategy pattern - classes | 21 |
| Table 3.3 Template Method pattern - classes | 22 |
| Table 4.1 Session Beans of the Query Augmentation Module | 33 |
| Table 4.2 Managed Beans and UI pages | 36 |
| Table 4.3 Navigation rules | 37 |
| Table 5.1 Configuration parameters | 47 |
| Table 6.1 Basic structure of test data | 52 |
| Table 6.2 Optimal number of search terms | 56 |
| Table 6.3 Search terms | 58 |
| Table 6.4 Yahoo Related Tags - Summary | 59 |
| Table 6.5 Yahoo Related Tags | 60 |

List of Source Code

| | |
|---|----|
| Code 3.1 Yahoo! Web Search result set..... | 18 |
| Code 3.2 HTML metadata..... | 18 |
| Code 3.3 Hook method..... | 23 |
| Code 3.4 Static class loading..... | 24 |
| Code 3.5 Dynamic class loading | 24 |
| Code 4.1 Entity Stopword | 34 |
| Code 4.2 Backing Bean class | 35 |
| Code 4.3 Configuration of a Managed Bean..... | 36 |
| Code 4.4 UI component tag..... | 36 |
| Code 4.5 Validation..... | 37 |
| Code 4.6 JSF datatable | 39 |

1

Introduction

| | |
|--|----------|
| 1.1 SOCRADES | 2 |
| 1.2 Application Service Catalogue | 4 |
| 1.3 Query Engine | 5 |
| 1.3.1 Why query augmentation?..... | 5 |
| 1.3.2 Query Augmentation Module | 6 |
| 1.4 Research questions | 6 |
| 1.5 Formal structure | 7 |
| 1.6 Conventions | 7 |

This master thesis is part of the SOCRADES project at SAP Research and it has been completed as a six-month internship at SAP Research CEC Zurich. Service-Oriented Cross-layer InFRAsTructure for Distributed smart Embedded deviceS (SOCRADES) is a European research and advanced development project. It is part of the Information Society Technologies (IST) initiative of the European Union's 6th Framework Programme (cf. [1]). 15 partners from 6 European countries are participating to SOCRADES. The major players of the industrial automation sector, Schneider Electric, ABB, and Siemens are leading the hardware development. SAP Research is responsible for the enterprise integration in backend business applications.

Without explaining more about SOCRADES we are presenting the SOCRADES integration architecture at SAP in Figure 1.1. The full integration architecture does not have to be understood but it just shows the dimensions of the project. SOCRADES at SAP Research is composed of several components and subcomponents over several layers. The Eventing component in the Application Interface layer can be used to subscribe to events of interest. The Service Monitor at the Service Management layer is monitoring services and managing information about services. And the Discovery component within the Monitoring and Inventory in the layer Device Management has listeners and a search engine to discover new services. There are lots of interactions between the components. The Eventing makes use of the Service Monitor and that in turn is accessing the Discovery component.

One to two team members of SOCRADES are working at each component. And this master thesis makes a contribution to the Application Service Catalogue at the top layer of the integration architecture.

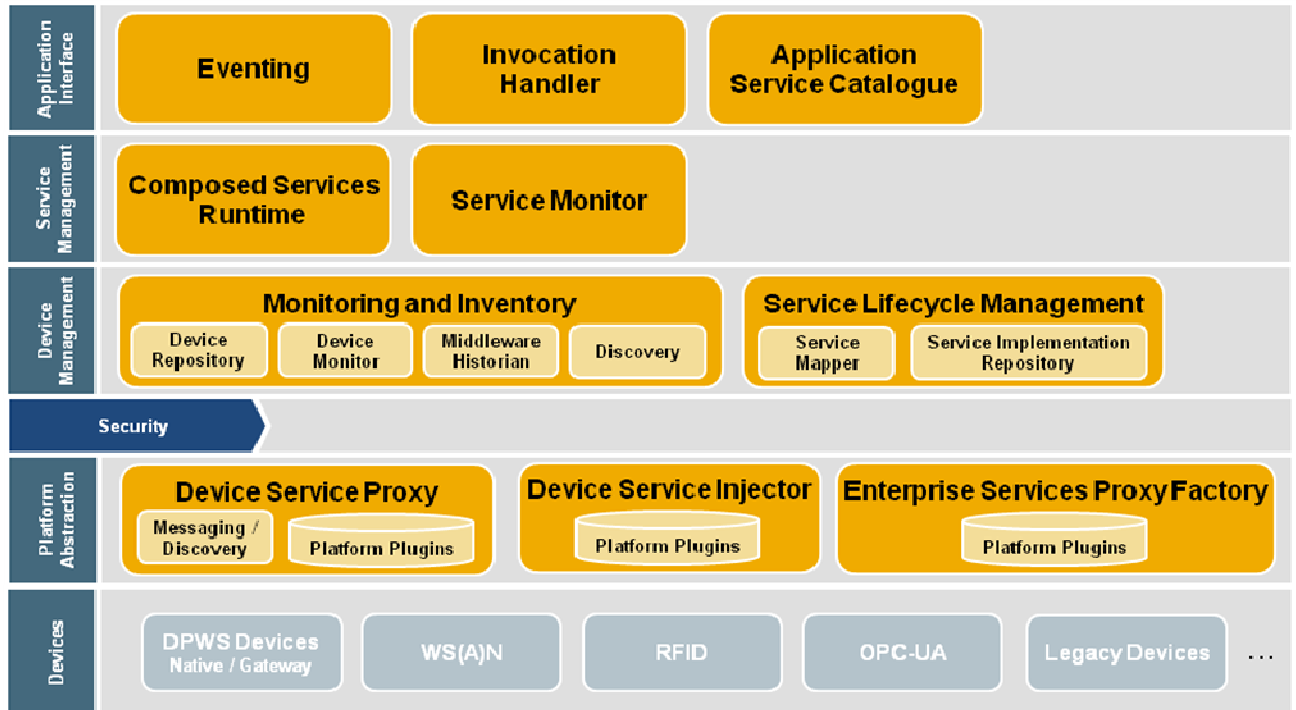


Figure 1.1 The SOCRADES integration architecture (taken from [1])

More precisely the development and evaluation of the subcomponent Query Engine of the Application Service Catalogue is the objective of this master thesis. Therefore, firstly, the introduction gives an overview about the SOCRADES project, secondly, describes the general idea of the Application Service Catalogue. Thirdly a simple outline of the proper objective of the Query Engine is given. The research questions of the thesis are following and finally the formal structure of the present report is summarized.

1.1 SOCRADES

SOCRADES middleware of SAP Research is closing the integration gap between shop floor and top floor. The shop floor is the production area of a factory at the operational level. The top floor is at the tactical and strategic level (enterprise level). Enterprise resource planning (ERP) is the software integration at top floor. Manufacturing Execution Systems (MES) are the software integration at shop floor.

Due to this gap enterprise business software has limited control of the shop floor. The integration architecture of SOCRADES is closing the gap and connects the production level with the enterprise software system of future manufacturing. In other terms SOCRADES middleware is the vertical integration from top to shop floor meaning ERP and MES are connected. The MES could even be skipped and the top floor is connected with the devices on the shop floor directly.

Service Oriented Architecture (SOA) concepts are used to connect smart objects at production level via embedded web services to the top floor. Devices are linked directly or via SOCRADES middleware. A smart object in this context is a general term for web service enabled devices, sensor networks, RFID devices and their applications.

Figure 1.2 illustrates the vertical integration from the top to the shop floor.

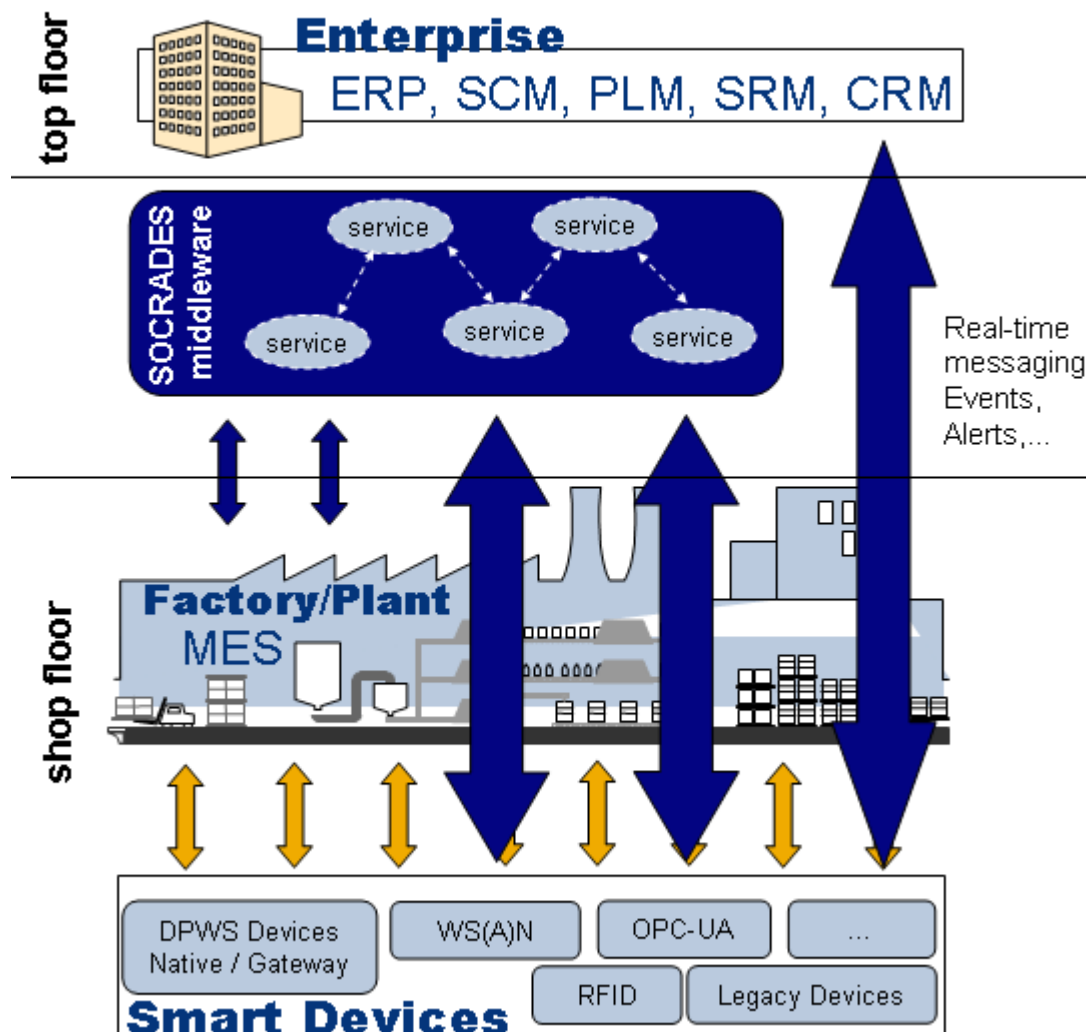


Figure 1.2 SOCRADES middleware (cf. [1])

The smart objects offering web services implement the Device Profile for Web Services (DPWS). DPWS is the web services implementation at the device level, which is based on SOAP. In addition two important features are defined in this standard: the web service discovery and web service eventing. Specification and other details about DPWS are available on [3].

At the top floor level an example of integration could be that a production machine has to be stopped because of overheating. This stop triggers an alert in form of a web service event. This alert is sent to the SOCRADES middleware and from there to the subscribers of the web service such as the ERP system. As soon as the ERP receives an alert like this it is recalculating the orders and informs the customers concerned about the estimated delays.

Coupling web service enabled devices with enterprise applications via SOCRADES has advantages such as service discovery, service catalogue, service life cycle management, device management and asynchronous communication.

The SOCRADES demo on YouTube [4] gives a very good overview of the project. And detailed information about SOCRADES is available on the project's website [1].

1.2 Application Service Catalogue

The Application Service Catalogue component is part of the top layer of the SOCRADES integration architecture. It is the entry point for enterprise applications. This component lets the user search for existing services. In addition methods for the discovery and injection of services are offered. The input is a simple text based query, which is expanded automatically as well as context knowledge such as location, time, and quality of service are added for the search.

The search process within the Application Service Catalogue is composed of three main steps. Each of these steps is executed in a subcomponent. In the Query Engine (Figure 1.3, subcomponent 1) a simple text query is enriched with context and semantics. (cf. [1]) Then, the best effort component (Figure 1.3, subcomponent 2) is rating the services. Finally, the on demand discovery and deployment component is used if no services have been found. Services are discovered dynamically or single services can be injected (Figure 1.3, subcomponent 3).

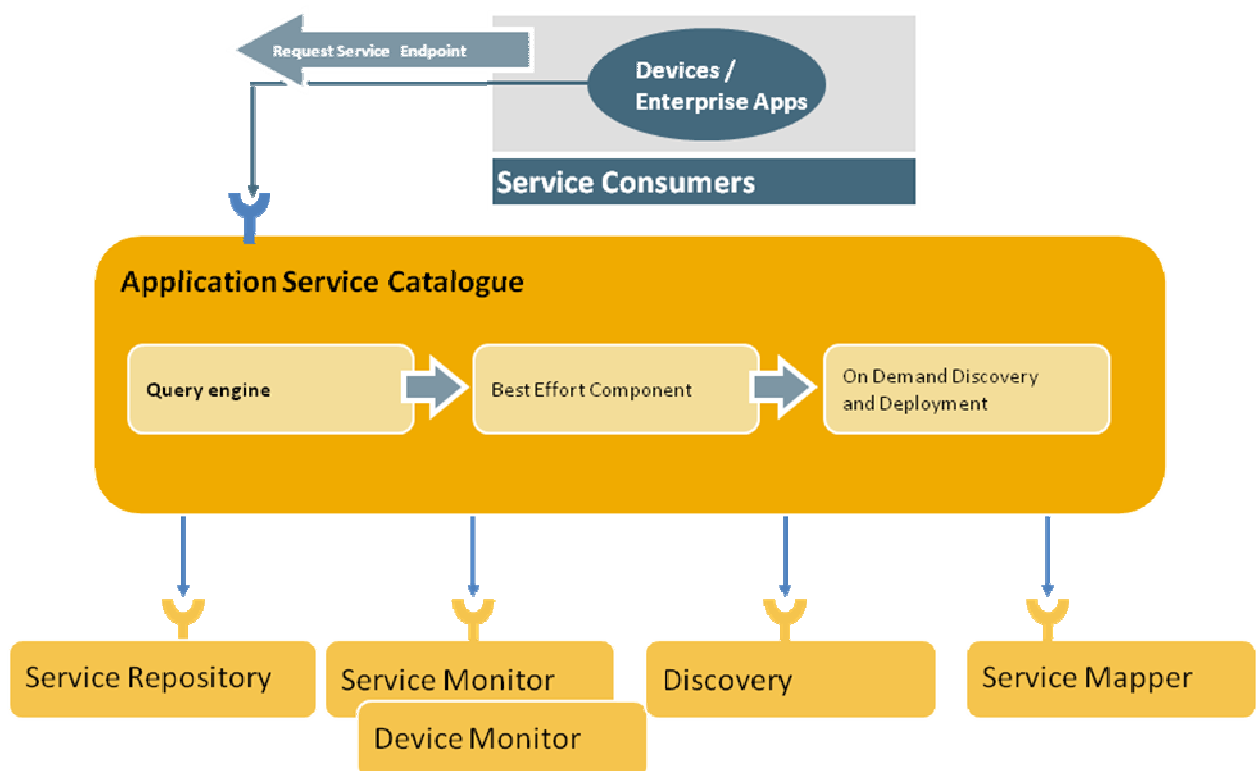


Figure 1.3 Application Service Catalogue overview [1]

The main scope of this master thesis is to design, implement and evaluate the architecture of the Query Engine. In addition to the proper specification of the component, a simple graphical user interface (GUI) for the Application Service Catalogue has to be developed. The GUI is visualizing the results and demonstrating the functionalities of the Application Service Catalogue especially the ones implemented for the master thesis.

After that short introduction to SOCRADES and the Application Service Catalogue we now focus on the Query Engine.

1.3 Query Engine

“The Query Engine is the core of the search. It consumes a simple text query and returns an ordered list of matching service types. In order to go beyond the simple query matching of a service name or description it first uses a concrete instance of a pluggable component called Query Augmentation.” [1] The goal of the Query Augmentation is to extend the simple text query before processing. This means when we input a simple text query, a so-called search term, and more expanded keywords are generated by the Query Augmentation, and then these keywords are submitted to query. The query usually consists of the initial search term and the expanded keywords.

Additionally, the Query Engine consumes a Query Context object, which contains information about the current location, activity and other preferences of its requester. The Query Context is added to the query before processing too.

Now we have been drilled down through the components and subcomponents of SOCRADES to the target component of this master thesis, the Query Augmentation component. It is also called Query Augmentation Module in the documentation.

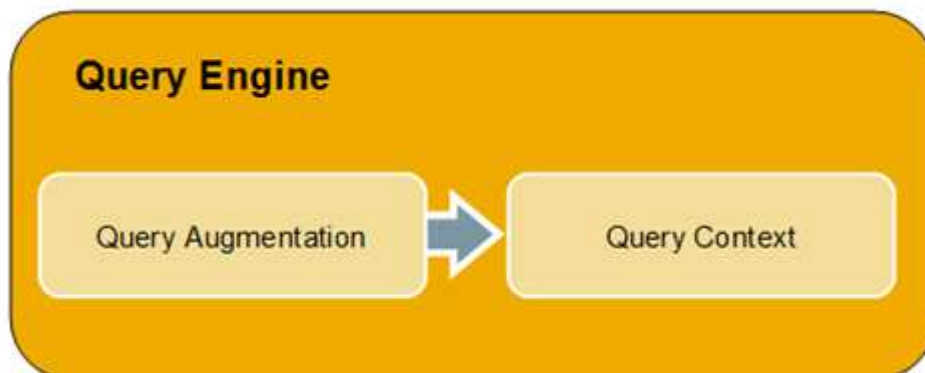


Figure 1.4 Query Engine

Before describing the Query Augmentation Module in detail, we are writing a little bit more about the background of this pluggable component.

1.3.1 Why query augmentation?

We know that production lines may have a lot of web service enabled devices and a backend business application may have to administrate several production lines at different locations. Therefore it is not possible to get a simple overview about the existing smart objects.

First, simple solution the Query Engine i.e., the Application Service Catalogue i.e., SOCRADES middleware offers, is the search for services by keywords.

Second, extended solution is enriching the initial keywords and adds context information to the query before processing the same search function as in the solution above. The enrichment

of the keywords is implemented in the Query Augmentation Module which is pluggable to the Query Engine.

Query enrichment, extension or augmentation means that related terms of the initial search term are added to the query. This raises the chance to find services instances. In order to understand the value added by the Query Augmentation Module read the following sample use case:

We assume that you are searching for services that are monitoring a boiler within your production line. Boiler means a container in which water is heated. You are starting the search for the service instances with the keyword boiler. The query boiler does not lead to the desired result and the monitoring services have not been found. The reason for that could be quite simple. The devices that are monitoring a boiler are described by their manufacturers as water meter. The search term and the desired result did not correspond with each other.

Using the Query Augmentation Module will raise your chances to find the monitoring services. You still set up your search with the search term boiler. The Query Strategy then enriches your search term by related keywords like water, steam, heat, and power before processing the search. This time you will find the desired monitoring services in the result.

1.3.2 Query Augmentation Module

The specification of the Query Augmentation Module demands an interface that supports several algorithms for the proper query enrichment. Thus the architecture of the component is based on the strategy design pattern [5]. The flexible design of the Query Augmentation Module allows to be extended by new strategies. Within the scope of my master thesis we developed several simple strategies and evaluated them.

- Wikipedia [6] is used as ontology generated and maintained by humans. The Wikipedia article of the initial search term of the search serves as a base to extend the search term with related keywords. Several versions of the Wikipedia strategy are implemented based on the words, links, and backlinks of an article.
- Yahoo Web Search API [7] represents a common search engine. Based on the search results two slightly different Yahoo strategies are offered.
- My Web 2.0 Web Services: Related Tags [8] is a Yahoo web service which seems to do exactly what we need for our Query Augmentation Module. A service that returns related tags.
- DBpedia [9] is a large multi-domain ontology which has been derived and extended from Wikipedia. This strategy demonstrates how semantic query enrichment can be implemented.

1.4 Research questions

The Query Augmentation Module is a pluggable component of the Query Engine. From this architecture the first research question follows:

1. Does Query Augmentation increase the result quality of the search for services on smart devices?

Considering the concrete implementations of strategies within this master thesis two other questions follow:

2. Do strategies based on human generated texts, such as a Wikipedia strategy, lead to better results than others, such as a strategy using a common search engine?
3. Which is the optimal strategy for the Query Augmentation of SOCRADES middleware?

1.5 Formal structure

This report is divided into seven Chapters, the content and the goals of each Chapter are explained here. The first Chapter, the introduction, gives a brief overview about the research project SOCRADES to which this master thesis belongs to. Introducing SOCRADES and setting limitations for the master thesis are the main objectives. Chapter 2 Related Work focuses on work related to query expansion strategies. Parts of this master thesis have already been defined in the specification of the SOCRADES integration architecture. So the related work treats the variable parts of the specification, which are the query augmentation in general and the individual strategies. The concrete ideas for strategies, their detailed design and the architecture of the query augmentation as one subcomponent of the SOCRADES project are documented in Chapter 3 Architecture. Then Chapter 4 Implementation describes the development tools and environment at SAP Research and compares them with other existing tools. In addition all the Java technologies that have been used for the implementation of the master thesis project are listed. After the implementation of the project it had to be evaluated in order to prove if SOCRADES benefits from the Query Augmentation Module or not. Chapter 5 gives introductions how the Query Augmentation User Interface (UI) has to be used and sums up the functionalities. Chapter 6 Experimental Evaluation contains the data collection, which has been done in the scope of a user study; the methodology used for the evaluation and of course the presentation and discussion of the results. The UI described in the previous Chapter has been implemented for that evaluation. The final Chapter 7 Conclusion contains some critical thoughts about the query augmentation how it is used in this master thesis and presents ideas for future improvements and extensions.

1.6 Conventions

This section describes the notations and conventions that are used throughout this report.

- The present report is composed of Chapters, that Chapters are divided into Sections which are further decomposed in Subsections. Where necessary, Subsections are broken down into paragraphs.
- **Bold** and *Italic* are used for emphasis and for highlighting terminology.
- `SansSerif` is used for URLs.
- `code` is used for all Java code and generally for anything that would be typed literally when programming.
- All resources are referenced in IEEE citation style link this [1].

- Figures, Tables and Code extracts are numbered inside a Chapter. For example, a reference to Figure *j* of Chapter *i* will be noted *Figure i.j*.
- Source code is displayed as follows:

```
1 package com.sap.sii.appServiceCatalog;  
2 ...  
3 public class QueryEngineManagedBean {  
4 ...
```

2

Related Work

| | |
|--------------------------------------|-----------|
| 2.1 Query Expansion | 9 |
| 2.2 Consuming Wikipedia | 10 |
| 2.3 Semantic Web..... | 12 |

The Service Application Catalogue of SOCRADES is offering a Search Engine in order to look for Web Services offered by Smart Devices. The main objective of this master thesis is the implementation and integration of the Query Augmentation Module, which helps to improve the Search Engine. Balke and Wagner show in [10] how common knowledge helps for the discovery and selection of Web Services. They considered user preferences and common knowledge of certain domains to improve the quality of service provisioning. The general idea of my master thesis has been given. As well as the general architecture and the development environment which were already defined within the specification of the SOCRADES integration architecture [11]. For the Query Augmentation one of the proposed concepts takes advantage from the web of knowledge that is the semi-structured information available on websites. The use of the web of knowledge ensures that, for instance, searching for an RFID reader would also return Auto-ID readers. Because of that the search for related work mainly focuses on work related to the algorithms for the Query Augmentation and their foundations. We call these algorithms query augmentation strategies. But before defining our own strategies we are looking for ideas and solutions to similar problems. They give us valuable inputs for our project.

One of the first ideas at the very beginning of this master thesis was to use Wikipedia as a large, multi-domain web of knowledge for the query augmentation strategies. Therefore we had a look at Wikipedia projects to see what already exists and what possibilities Wikipedia really presents to us. The main focus of this Chapter is therefore put on Wikipedia projects. But let us start with a short Section about query expansion in general.

2.1 Query Expansion

Query expansion is a process of enhancing original query formulations in information retrieval. We consider consuming online resources by accessing public web services such as MediaWiki API [6], Yahoo Web Search [7] or Google AJAX Search API [12] for the Query Augmentation of SOCRADES. Usually original queries are short and under-specified therefore enhanced terms help to improve the retrieval of services from the SOCRADES

Application Service Catalogue. In the context of our search engine, query expansion means evaluating a user's input and expanding the search query to match additional services (cf. [13]). Finding techniques for the query expansion, also called Query Augmentation for the Query Engine, is the goal of the study of the related work.

The paper "Query Expansion Based on a Personalized Web Search Model" [14] is proposing a query expansion algorithm. The suggested solution is a middleware between the user and the web search engine. The middleware can learn a user's preferences implicitly, generates a user profile and uses it for finding more personalized expansion words, which are then added to the query before submitting it. A user's profile is generated on the basis of the user's visited web resources.

The Query Augmentation of SOCRADES can be regarded as middleware too. It is a system between the user and the actual search engine for web services. So far the Query Augmentation is not intended to be personalized but such an approach could definitively be used to optimize SOCRADES. Personalization should be checked on different levels. Maybe it is not necessary to generate a user profile per user but rather per user group or even only one per system. Such a query expansion is supporting the user's search intention. Per default SOCRADES must not work domain specific. By introducing a profile per user group or system for the query expansion SOCRADES could become aware of the different intentions, context, industry or general surrounding of the use. In contrast to the Personalized Search System [14] we do not expect different results to different users, but the optimal result within the whole system. That also strengthens the idea of introducing one profile per system in order to differ from other uses.

According to [14] it is necessary to determine the number of expansion words. Too many search terms might bring in noise and a search engine returns many irrelevant results. That is a very important point for all the techniques of query expansion and it is taken into consideration in our work too. The last interesting input we found in [14] is the fact, that there were no resources for large scale evaluations but instead the proof of concept was made by experiments. To sum up the Personalized Search System [14] is a broad query expansion system with lots of inputs to our Query Augmentation in general. The query expansion presented in [14] is based on the web history of a user and therefore personalized. By contrast we are thinking about a more general query expansion and we want to use Wikipedia as a foundation, which is treated in the next Section.

2.2 Consuming Wikipedia

Considering the words that appear in a Wikipedia article as a foundation of the Query Augmentation is the basic concept of our approach. We have chosen Wikipedia because it is not domain specific, contains a lot of articles and has an active community. With related projects we can prove the helpfulness of Wikipedia. And in addition we describe different solutions to access Wikipedia.

The research on Wikipedia is very active and a lot of research work exists. Here is a selection of works, which gave important input to our work. Bachlechner and Siorpaes [15] are using Wikipedia entries as ontology elements and show that URIs of Wikipedia entries are reliable identifiers for ontology concepts. The identifiers of the articles are the URIs. Wikipedia may be the largest existing ontology. Currently the English version of Wikipedia has more than 2'653'000 entries, which means it holds unique identifiers for more than 2'653'000 concepts.

The Wikipedia community, which is maintaining the content, seems to work very consistent. According to [15] only 5% of the concepts/articles change in a major sense during their lifespan. So the Wikipedia can be easily used as ontology without modification. But difficulties with Wikipedia are documented in [15] too. One of the problems is that certain terms have several meanings. If one is then referring to wrong concept the intentions of the user are misunderstood, which can lead to incomplete and/or inconstant results.

“Wikipedia as an Ontology for Describing Documents” [16] is using Wikipedia articles to predict concepts common to a set of documents. Several algorithms are described to aggregate and refine the result. The experiments of that work show that it is possible to predict concepts common to a set of documents by using the Wikipedia article text and links. “One general approach to describing what a document is about is to use statistical techniques to describe the words and phrases it contains.” Another approach could be to tag the document with relevant terms that represent semantic concepts such as Wikipedia articles. The tags correspond to the URI of a Wikipedia article. The Wikipedia entries represent a consensus view of a community of users, they are maintained by a community and they can be accessed for free. And hopefully the Wikipedia articles deliver detailed descriptions of a document. All this are benefits of Wikipedia mentioned in [16].

Consequently Wikipedia seems to be a good foundation for improving search results. But how can we extract the useful and interesting information from Wikipedia articles. One possible answer is presented by Google. Google definitively found successful search strategies. And the main features of Google are used within our augmentation strategies too. The fundamental ideas behind Google are presented in [17] by Brin and Page. Google is querying the web, which heavy uses structured hypertext and which is growing rapidly. Wikipedia meets these two requirements too. Brin and Page say that relying on keyword matching usually returns too many low quality matches for a search request. So Google makes use of link structure and text that provide a lot of information for making relevance judgments and quality filtering. Google makes use of the link structure of the Web to calculate a quality ranking a web page. This ranking is called PageRank. In addition Google utilizes links to improve search results. The PageRank is calculated by counting citations or backlinks of a page. The analysis of links is not only used by Google. In [16] Syed et al. did not only work with Wikipedia articles but experimented with the article links too. So the analysis of links is definitively a point we kept for our Query Augmentation. The strategies should not only be based on Wikipedia articles but maybe on links, backlinks and other structured expressions within the articles.

Another very similar problem has been solved by a team of the HP Laboratories in India. They successfully demonstrated the value of using Wikipedia as a source for enriching content [18]. The clustering system of [18] uses Wikipedia to categorize news feeds. The news feeds themselves contain only a few words with minimal information. Wikipedia is used to enrich the content and then summarizing similar news feeds. So it is another project that is adding value to its initial information using Wikipedia. As all the Wikipedia projects we evaluated the system developed in [18] is based on a database dump of Wikipedia and thus works with an offline version of Wikipedia. We rather decided to use a more dynamic version by connecting to the live version of Wikipedia whenever possible. That way we can profit from the latest entries and ameliorations. Nevertheless, there is an interesting point about the offline version of Wikipedia as used in [18]: the downloaded dump has been cleaned before usage. That means that articles such as templates, very short articles and articles describing Wikipedia itself have been removed before using the system for the content enrichment. Such a cleaning or filtering of articles is considered within our work too.

2.3 Semantic Web

We are not only thinking about traditional web resources and web search engines but also consider newer web technologies such as the semantic web. With the Semantic Web web pages are understood no longer only from humans, but in certain ways from machines too. New structured and formalised information becomes usable for automatic processing. Today's search engines do not know the semantics of user inputs and web pages and the semantic web and semantic search engines want to overcome that limitation.

Established technologies in the semantic Web are RDF (Resource Description Framework) [19], OWL (Web Ontology Language) [20] and microformats [21]. While one can make structured information available with RDF, OWL describes the ontology, provides meta information and links these together. SPARQL [22] is the query language for RDF and thus for the Semantic Web. As one can guess by its name, there is a certain similarity to SQL, but the function range is smaller and more limited. We can search within the semantic web as in the “normal” web. We just need other tools do to that. Tools, examples and lots of works on that subject can be found on the project web page of DBpedia [9].

The DBpedia project [9] is the Wikipedia project of the Semantic Web we are describing now. “DBpedia is a community effort to extract structured information from Wikipedia and to make this information available on the Web. DBpedia allows you to ask sophisticated queries against Wikipedia, and to link other data sets on the Web to Wikipedia data.” [9] The structured information that has been extracted from Wikipedia combines the information into a huge, cross-domain knowledge base called DBpedia data set. In order to identify semantic concepts DBpedia uses the URI of Wikipedia articles too. A Wikipedia URI (e.g. <http://en.wikipedia.org/wiki/RFID>) becomes a DBpedia resource URI (e.g. <http://dbpedia.org/resource/RFID>). DBpedia is not only identifying things, the more important task of DBpedia is to describe things via Wikipedia article, Wikipedia categories and YAGO [23]. YAGO is a huge semantic knowledge base that knows more than 2 million entities (like persons, organizations, cities, etc.).

“DBpedia uses the Resource Description Framework (RDF) as a flexible data model for representing extracted information and for publishing it on the Web. We use the SPARQL query language to query this data.”[9] The project team is offering an online access [24], with limited data sets loaded, hosted on an OpenLink Virtuoso server [25]. Datasets can also be downloaded and installed on an own back-end database engine. In contrast to the real Wikipedia the following information and descriptions of Wikipedia articles are available in datasets. Here is a list of the core datasets [9] that could be interesting for the Query Augmentation of the Application Service Catalogue:

- Titles; titles of all Wikipedia Articles in the corresponding language.
- Short abstracts; Short Abstracts (max. 500 chars long) of Wikipedia Articles.
- Extended abstracts; Additional, extended English abstracts (max. 3000 chars long).
- Links to Wikipedia article; Links to corresponding Articles in Wikipedia.
- Article categories; Links from concepts to categories using the W3C’s Simple Knowledge Organization System (SKOS) [26] vocabulary.
- Pagelinks; Dataset containing internal links between DBpedia instances. The dataset was created from the internal pagelinks between Wikipedia articles. The dataset might

be useful for structural analysis, data mining or for ranking DBpedia instances using Page Rank or similar algorithms.

- Redirects; Dataset containing redirects between Articles in Wikipedia.
- Disambiguation links; Extraction from Disambiguation Templates.

Checking this list, DBpedia seems to be a source with a lot of potential for the Query Augmentation.

3

Architecture

| | |
|--|-----------|
| 3.1 Query strategies | 15 |
| 3.1.1 Wikipedia Strategy | 17 |
| 3.1.2 Yahoo Strategy | 17 |
| 3.1.3 DBpedia Strategy | 19 |
| 3.2 Query Augmentation Module..... | 19 |
| 3.2.1 Strategy design pattern | 20 |
| 3.2.2 Template method design pattern..... | 21 |
| 3.2.3 Dynamic class loading | 23 |
| 3.2.4 Architectural overview of the Query Augmentation | 25 |
| 3.3 Query Augmentation GUI | 26 |
| 3.4 Future Work | 26 |
| 3.4.1 Tagging | 26 |

This Chapter documents the most important facts about the software architecture of this master thesis. It is mainly describing how the ideas collected in Chapter 2 Related Work are realized in the Query Augmentation Module.

Figure 3.1 gives an overview of the components that have to be implemented in the scope of this master thesis. Basically it is a traditional 3-layer client server software architecture in the web environment. At the top there is the Query Augmentation GUI. On the Java EE Server below is the Query Augmentation Module. The core task of the present master thesis is the development of the concrete query strategies. Therefore the main focus of this Chapter is put on the concept and design of the query strategies.

It is not possible to treat the complete design details of the Query Augmentation Module in this report. We refer to the source code and the Javadoc of the Query Engine component (available on [27]) that documents the work in detail.

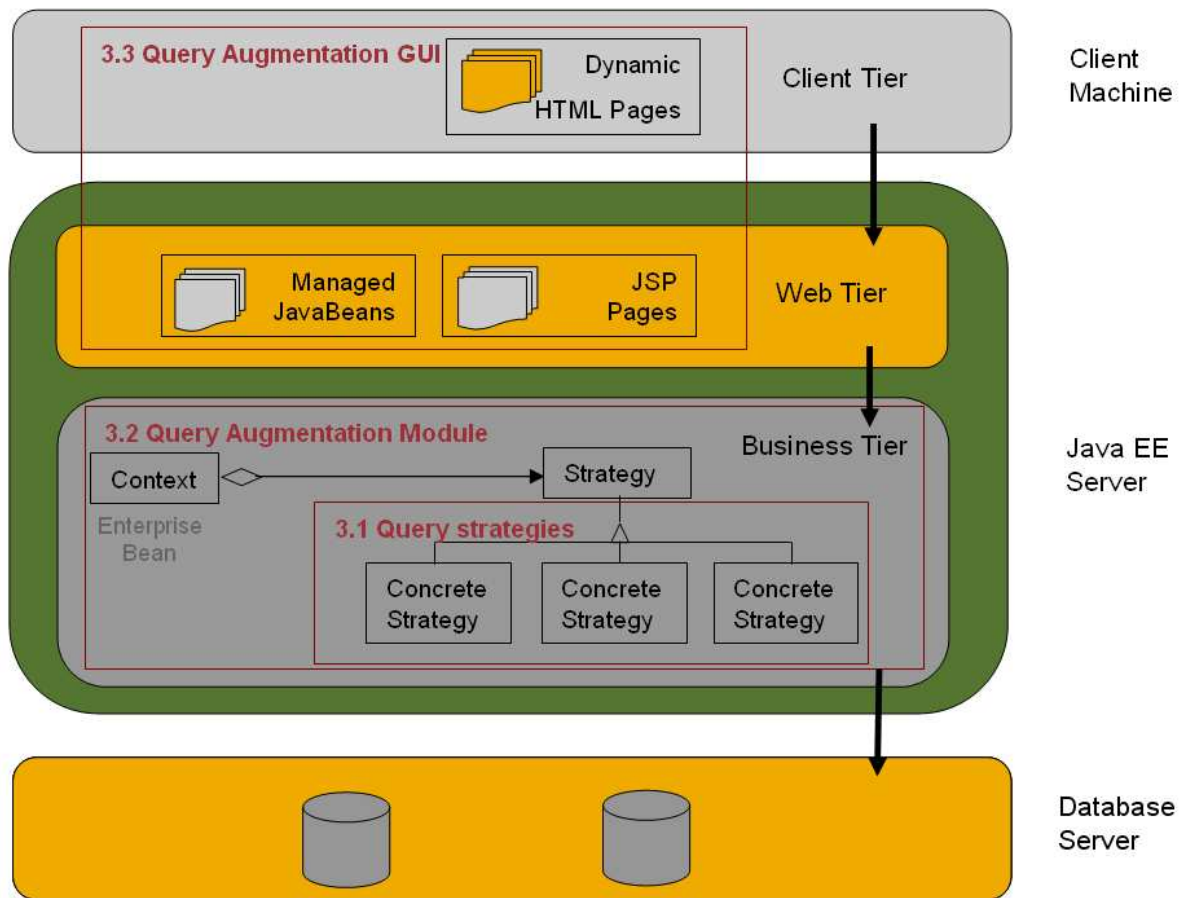


Figure 3.1 Architectural overview in style of Figure 1-1 of [28].

Chapter 1 Introduction gave a general overview of the architecture based on a top down approach (1.1 SOCRADES, 1.2 Application Service Catalogue, and 1.3 Query Engine). Contrary to the introduction this Chapter is built up on a bottom up approach. In order to understand the architecture of the master thesis the low level details and leading thoughts are documented first in Section 3.1 Query strategies. The second Section 3.2 Query Augmentation Module describes how we brought together the different strategies within one component in terms of software architecture. The third Section describes the ideas of the Query Augmentation GUI. And finally the Section 3.4 Future Work lists ideas for future improvements of the component, which came up during the development of the Query Augmentation Module but could not be included directly.

3.1 Query strategies

The strategies for query enrichment are the foundation of this master thesis and they are kept general as SOCRADES is not a domain specific implementation by default. According to the specification [1] it is planned to use the SOCRADES integration architecture in automation, energy and insurance industry.

This Section is answering the question how the web resources are accessed and treated within the Query Augmentation. As mentioned in 1.3.2 Query Augmentation Module we decided to develop query expansions strategies on the base of the Wikipedia [6], Yahoo Web Search [7], and DBpedia [9]. Among the common search engines Yahoo! has been chosen because of the

offered API. Yahoo Web Search is the largest web search engine offering a REST interface and returning a XML document. A REST interface allows a simple integration into Java business applications with little effort. In contrast to Yahoo! Google offers the Google AJAX Search API, which is RESTful but returns JSON format [29] only. And Microsoft's live search features a SOAP API.

The foundation of the query strategies is defined and the next step that has to be defined is the treatment of the web resources. How can useful information be extracted and enrich a query? In order to keep the method simple the basics of **text analysis** have been examined. One sort of text analysis is the statistical one. This means counting particular features of the textual data and then applying mathematical transformations. According to [30] the simplest type of statistical text analysis produces frequency lists of word-forms, usually arranged from the most to the least frequent. The frequency of word-forms is only roughly related to what a text says, but it is related, and so it is worth to work with. This text analysis method is part of the Query Augmentation Module and in the context of this work it is called **Word Frequency Counter**. The Word Frequency Counter of the Query Augmentation Module works with a user defined stop word list. This list is composed of words that have to be ignored in the frequency list of word-forms. 'Is', 'are', 'and', 'or', 'all', and a lot of other common words are part of the stop word list, as they are quiet frequent in an article but do not have a relation to the query.

Consequently the general method of enriching a query could be described like this: A user defines a simple query e.g. "Temperature" in the Application Service Catalogue. The sub component Query Augmentation Module requests the defined web resources (Yahoo! Web Search, Wikipedia and/or DBpedia) for the query, analyzes the response (web search result or Wikipedia article) with the Word Frequency Counter, and finally enriches the query with the most frequent word-forms found. The enriched query is then used from the Application Service Catalogue to find available Web Services within the system.

The described method builds the frame for all query strategies. From the coding perspective the following steps have to be implemented fro each strategy:

1. Compose the request URL. *E.g. compose an URL for querying the MediaWiki API in order to get the latest revision of the Wikipedia article "Temperature"*.
2. Execute the web request and get the data from the web. *E.g. get the search result of the Yahoo! Web Search for the query "Temperature" in the format of an XML document.*
3. Extract the useful information, text, or other data from the answer of the web request.
4. Clean up the extracted data. *E.g. remove the punctuation as it is not relevant for the Word Frequency Counter.*
5. Split the cleaned text into an array of single words.
6. Calculate the frequency of each word in the text with the Word Frequency Counter and return the top ones, which represent the augmented keywords.

The three sources Wikipedia, Yahoo! and DBpedia lead to the three main query augmentation strategies of the Query Augmentation Module documented in the following Subsections: 3.1.1 Wikipedia Strategy, 3.1.2 Yahoo Strategy, and 3.1.3 DBpedia Strategy. During the analysis and design of the single strategies some extensions and alternatives have been discovered and out of the three main strategies eight concrete augmentation strategies have been developed in the Query Augmentation Module:

- 4 Wikipedia strategies,
- 3 Yahoo! Web Search strategies, and
- 1 DBpedia strategy.

The strategies, the derivatives and the special features of each strategy are the subject of the following Subsections.

3.1.1 Wikipedia Strategy

At the beginning the Wikipedia Strategy focused on the words of the article only. While processing some tests and analysing the articles it became clear that Wikipedia articles contain more information for the query enrichment. As Google [17] the Wikipedia Strategy could make use of links, backlinks, and formatting details within the text analysis.

Consequently several algorithms build upon the Wikipedia Strategy. Here's a short description of the ideas behind the basic strategy and all derived ones. The input of the Word Frequency Counter differs between the single Wikipedia strategies.

| Wikipedia Strategy | The Word Frequency Counter is applied on... |
|---------------------------------|--|
| WikipediaStrategy | the complete Wikipedia article. |
| WikipediaStrategyLinks | the links of a Wikipedia article. |
| WikipediaStrategyBacklinks | the backlinks of a Wikipedia article. |
| WikipediaStrategyBoldAndItalics | the bold and italic words of a Wikipedia article. |

Table 3.1 Wikipedia strategies

3.1.2 Yahoo Strategy

Querying the Yahoo! Web Search with the initial query the Yahoo! REST API returns a result set that contains the information for the query augmentation. E.g. this URL requests the top 10 results for the query "Temperature":

<http://search.yahooapis.com/WebSearchService/V1/webSearch?appid=YahooDemo&query=Temperature&results=10>

The result set is always returned in a XML document structured like Code 3.1. The basic Yahoo Strategy (**YahooStrategy**) of the Query Augmentation Module is using the collection of the Summary elements as input of for the Word Frequency Counter.

```

1   <ResultSet xsi:schemaLocation="urn:yahoo:srch
    http://api.search.yahoo.com/WebSearchService/V1/WebSearchRespon
    se.xsd"
2           type="web"
3           totalResultsAvailable="707000000"
4           totalResultsReturned="1"
5           firstResultPosition="1"
6
    moreSearch="/WebSearchService/V1/webSearch?query=Temperature&app
    id=YahooDemo&region=us">
7   <Result>
8       <Title>temperature: Definition from Answers.com</Title>
9       <Summary>
10          temperature n. The degree of hotness or coldness of a body
            or environment. ... Thermometers measure temperature by a number
            of means, including the expansion ...
11          </Summary>
12          <Url>http://www.answers.com/topic/temperature</Url>
13          ...
14      </Result>
15      ...
16  </ResultSet>

```

Code 3.1 Yahoo! Web Search result set

Compared to the extensive articles of Wikipedia the summaries of the Yahoo result sets seem to deliver less information. Because of that a derivate of the Yahoo Strategy has been developed. The strategy is called **YahooStrategyKeywords** and it continues processing the result set of Yahoo instead of using it for the text analysis. The strategy is also called “Yahoo – Extracted Metadata Of Top Results” - Strategy and that already explains the idea behind it. This concrete strategy makes use of the Url element of the Web Search result set and continues processing the URLs. This Yahoo Strategy is requesting the web resources of the URLs and extracts the metadata description and keywords from it. The collection of all the descriptions and keywords then builds the input of the Word Frequency Counter of the Query Augmentation.

E.g. Code 3.1 contains the Url <http://www.answers.com/topic/temperature> in the result set, which in turn contains the metadata illustrated in Code 3.2.

```

1   <html>
2   <head>
3   ...
4   <meta name="description" content="temperature: Web Search
    Results from Answers.com" >
5   <meta name="keywords" content="temperature, Information,
    reference">
6   ...
7   </head>

```

Code 3.2 HTML metadata

Yahoo Related Tags strategy

Besides the common web search Yahoo is offering lots of other web services. The Web Service Yahoo Related Tags [8] seemed to merge perfectly with the idea of the query augmentation within SOCRADES' Application Service Catalogue. This Yahoo! Web Service is retuning related tags for a query. These related tags could serve as the keywords of the enriched query.

For the implementation the steps with the text analysis, text preparation and the Word Frequency Counter within the augmentation algorithm can be neglected. Therefore this strategy causes the least work within the Query Augmentation Module.

3.1.3 DBpedia Strategy

“The DBpedia data set is a large multi-domain ontology which has been derived from Wikipedia.” [9] Basically DBpedia is another interface for accessing Wikipedia articles.

DBpedia differs from MediaWiki API in the following points:

- DBpedia is a project of the Semantic Web.
- DBpedia is based on a database dump of Wikipedia.
- Wikipedia is enriched with the DBpedia ontology.
- DBpedia can be accessed via the Jena framework [31] that is a framework for Semantic Web applications.

DBpedia offers a lot of additional internal and added external information about Wikipedia articles that cannot be extracted from Wikipedia directly. The DBpedia Strategy of the Query Augmentation Module does not yet benefit from the advantages of the Semantic Web and the DBpedia ontology. But this strategy provides a proof of concept, that the Wikipedia can also be integrated via a Semantic Web application.

3.2 Query Augmentation Module

The previous introduced query augmentation strategies have to be brought together in the Query Augmentation Module. The main requirements of the component are that it has to be **extensible** and **pluggable**. Extensible means that the component has to offer a simple interface for the integration of additional strategies. Pluggable in this context means that the Application Service Catalogue has to work with and without the Query Augmentation Module. If these two requirements are respected the Query Augmentation Module does not lock the architecture with a search method that is not appropriated in a running instance of the SOCRADES architecture.

The specification of SOCRADES integration architecture [11] requires the use of the strategy design pattern [5] for the Query Augmentation Module. After the definition of the general query augmentation algorithm in 3.1 Query strategies the template method design pattern [5] seems to be appropriate. These two patterns define the rough structure of the Query Augmentation Module. The following Subsections are devoted to these patterns and other relevant principles of the object oriented programming.

3.2.1 Strategy design pattern

The strategy pattern belongs to the behavioural design patterns. E. Gamma et al. define the intent of the pattern in [5] as following. “Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.” The definition represents the requirements of the Query Augmentation Module perfectly. The Query Engine component has to be able to call its subcomponent Query Augmentation for the enrichment of a query with a certain, interchangeable strategy.

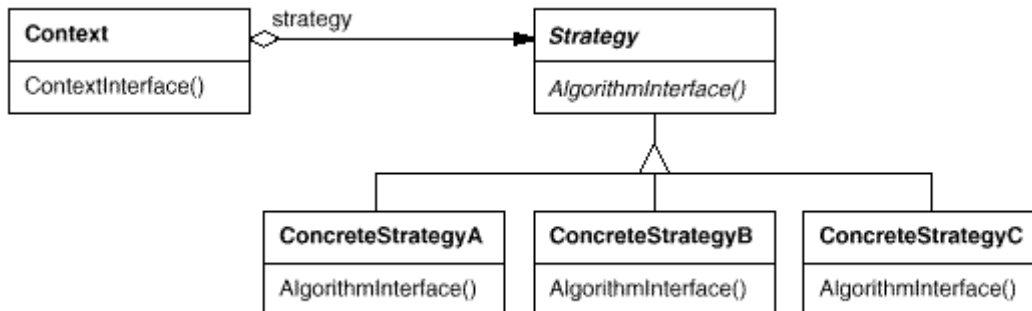


Figure 3.2 Strategy design pattern [5]

Strategy declares an interface common to all supported algorithms. The **Context** uses this interface to call the algorithm defined by a **ConcreteStrategy**. A **ConcreteStrategy** implements the algorithm using the **Strategy** interface.

Table 3.1 lists the java classes of the Query Augmentation Module that are part of the strategy pattern. A class diagram of the component is displayed at the end of this Subsection on Figure 3.5.

| Participant of the design pattern | Java class of the Query Augmentation Module | Comment |
|-----------------------------------|---|---|
| Strategy | Strategy | The interface offers one method called 'getEnrichedKeywords' which returns the enriched input query. |
| ConcreteStrategy | <i>ConcreteStrategy</i> YahooStrategy YahooStrategyKeywords YahooStrategyRelatedTags DBpediaStrategy WikipediaStrategy WikipediaStrategyLinks WikipediaStrategyBacklinks WikipediaStrategyBoldAndItalic | <i>ConcreteStrategy</i> is the only ConcreteStrategy that directly implements the interface, it is the implementation of the general query augmentation algorithm, and therefore it is an abstract class. |

| Participant of the design pattern | Java class of the Query Augmentation Module | Comment |
|-----------------------------------|---|---|
| Context | StrategyContext | Exposed a WebService and JavaBean in order to be called from outside. |

Table 3.2 Strategy pattern - classes

Two questions rose with the application of the strategy pattern:

1. How can the common parts of the query augmentation strategies be integrated without writing redundant code?

As already mentioned the general query augmentation algorithm can be defined within a Template Method. The steps of the algorithm could also be in the Strategy interface. But as the sequence of the method calls is the same for each strategy it would be smart if they are integrated into a Template Method, which is documented in the next Subsection 3.2.2 Template method design pattern.

2. How can the concrete strategies be called from the Context dynamically, that the Context does not have to be touched if concrete strategies are added to the Query Augmentation Module?

This question is answered in the Subsection 3.2.3 Dynamic class loading.

3.2.2 Template method design pattern

Template Method is often used for defining the algorithm's steps of a Strategy (cf. [5]) and so it is in the case of the Query Augmentation Module too. According to E. Gamma et al. [5] the Template Method defines the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

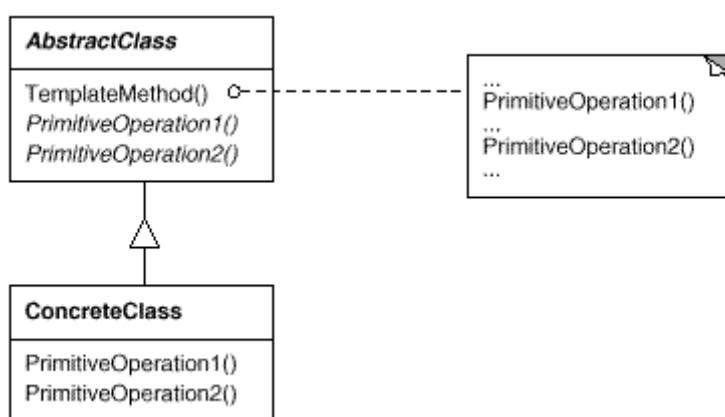


Figure 3.3 Template Method design pattern [5]

The **AbstractClass** of the Query Augmentation Module is the ConcreteStrategy that implements the interface of the Strategy pattern. The Template Method and the Primitive Operations of the Query Augmentation Module are listed in the following table:

| AbstractClass | Java method name | Comment |
|---------------------|---------------------|---|
| TemplateMethod | getEnrichedKeywords | Implemented from Strategy interface. The main template method basically implements the 6 basic steps described at page 16. |
| PrimitiveOperation1 | getURL | Composing the request URL. |
| PrimitiveOperation2 | getDataFromWeb | Executing the web request and get data from web. |
| PrimitiveOperation3 | getSummary | Extract useful information. |
| PrimitiveOperation4 | getCleanedSummary | Clean up the extracted data. |
| PrimitiveOperation5 | getStringList | Split the text into an array of single words. |
| PrimitiveOperation6 | getKeywords | Get the most frequent word-forms from the text. |

Table 3.3 Template Method pattern - classes

Some of the algorithm's steps are common within a majority of the concrete strategies and therefore the template method is not only used for the definition of the Primitive Operations but offers a default implementation of them too.

If a default implementation cannot be used within a certain strategy it can always be overwritten in the derived class. An overwritten method always means a complete different implementation of a certain step of the query augmentation algorithm. In the case of the method `getSummary`, which is extracting the useful information of a web request response, has a general and a strategy specific part. That complexity can be solved with hook methods.

Hook methods

A hook method is a method which is called from a `TemplateMethod`. The functionality of the `TemplateMethod` is shifted to the `HookMethod` to allow customisation of the `TemplateMethod`'s functionality by later (e.g. in a derived class) changing the hook method (cf. [32]).

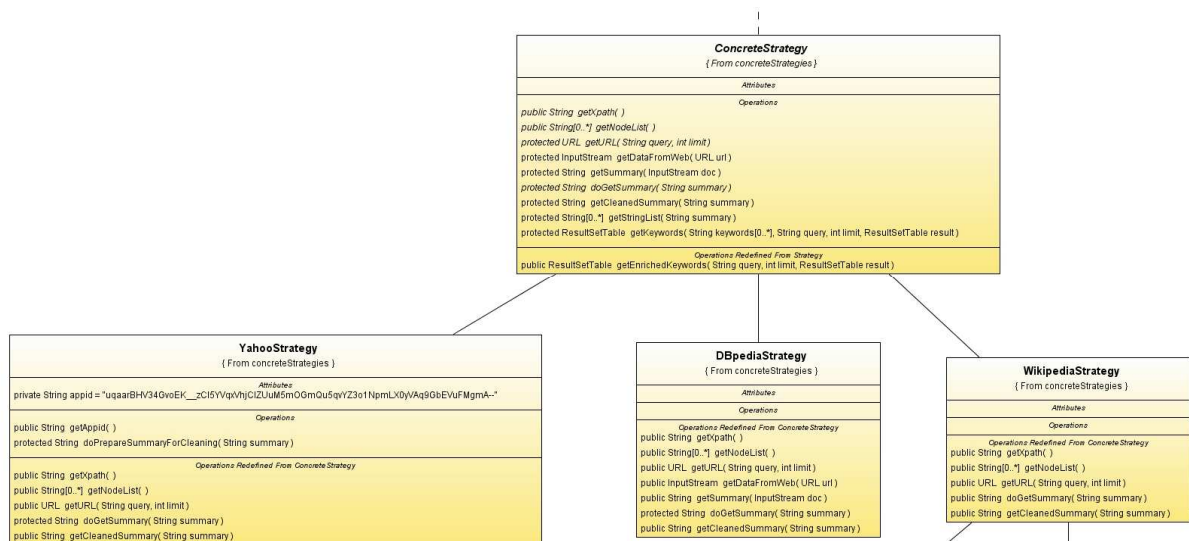
```

1   public abstract class ConcreteStrategy {
2       public String getSummary(InputStream doc) {
3           // template method
4           ...
5           summary = doGetSummary(summary)
6           // call hook method
7       }
8
9       public String doGetSummary(String summary) {
10          // default implementation of the hook method
11      }
12  }
13
14  public class WikipediaStrategy extends ConcreteStrategy {
15      public String doGetSummary(String summary) {
16          // refined implementation of the hook method
17      }
18  }

```

Code 3.3 Hook method

An example of a hook method with the Query Augmentation Module is the method `getSummary` and its hook method `doGetSummary`.

Figure 3.4 The hook method `doGetSummary`

Explanation of Figure 3.4: `ConcreteStrategy` implements the general `getSummary` method and offers the hotspot `doGetSummary` for the customisation of the `getSummary`. `YahooStrategy` and `WikipediaStrategy` make use of this hotspot and implement the hook method. `DBpediaStrategy` has a complete different `getSummary` method and is rewriting it.

3.2.3 Dynamic class loading

Within the Context of the Strategy pattern the following question rose: How can concrete strategies be called dynamically so that the Context does not have to be touched if strategies are added to the Query Augmentation Module?

The Context of the Query Augmentation Module, the `StrategyContext`, is the interface to the world outside the component. In order to be able to use the Query Augmentation Module the interface of the Context and the name of the available concrete strategies have to be known. In a first approach an array of valid strategy names has been published. The strategy names have been short and easy to remember e.g. “wikipedia” for the basic Wikipedia Strategy.

Within the Context the concrete strategies have been loaded with if – else statements:

```
1   if (strategy == "wikipedia")
2       Strategy s = new WikipediaStrategy()
3   else if ...
```

Code 3.4 Static class loading

If somebody wants to a new concrete strategy to the Query Augmentation Module a new strategy name has to be published and the Context has to be updated with an additional else-if case.

This problem leads us to the dynamic class loading within the Context as documented in Code 3.5. Instead of delivering a short strategy name to the Context, the Context now expects a class name of an existing strategy (e.g. `com.sap.sii.appServiceCatalog.queryEngine.strategy.concreteStrategies.WikipediaStrategy`).

For the human the class names are more complicated to remember, but for the flexibility of the component the dynamic class loading is a more adequate solution.

```
1   // create the strategy instance of the defined strategy type.
2   Class c = Class.forName(strategyClass);
3   Strategy strategy = (Strategy) c.newInstance();
4   // call the getEnrichedKeywords of the defined strategy type.
5   ResultSetTable result = strategy.getEnrichedKeywords(query,
    limit, result);
```

Code 3.5 Dynamic class loading

3.2.4 Architectural overview of the Query Augmentation

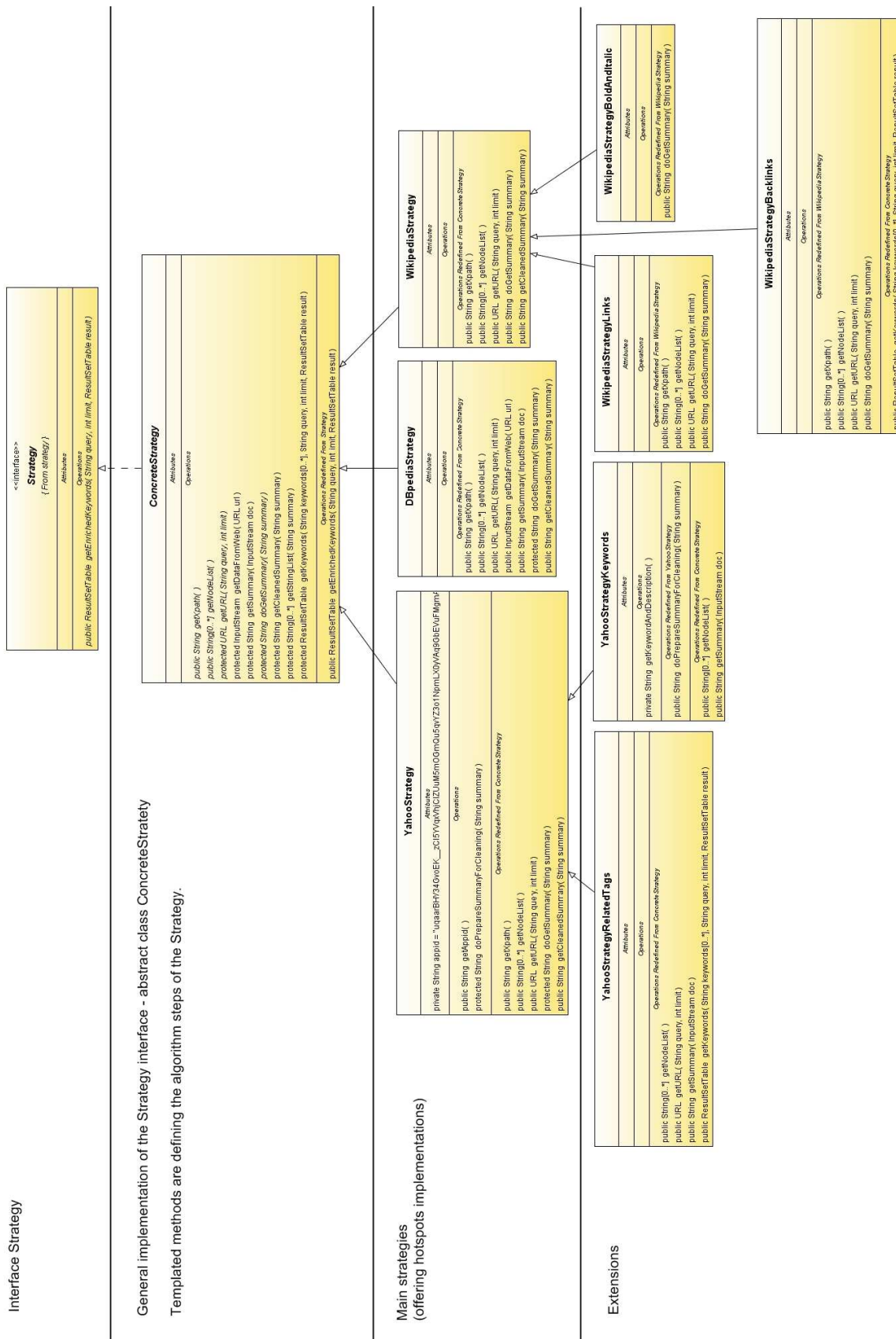


Figure 3.5 Architectural overview of the Query Augmentation

3.3 Query Augmentation GUI

The Query Augmentation Module is part of the Query Engine, which is a subcomponent of the Application Service Catalogue that in turn is a component of SOCRADES (cf. 1 Introduction). Within the scope of the present master thesis the integration of the Query Augmentation Module is guaranteed but not fulfilled. The integration is provided via Web Service offered by the Query Augmentation Module in the Strategy Context. In order to be independent of other components of SOCRADES a simple Query Augmentation GUI has been developed. Technical details of this GUI are discussed in Section 4.2 Java EE of Chapter 4 Implementation and a detailed description is given in Chapter 5 User manual.

3.4 Future Work

During the implementation and evaluation of the Query Augmentation Module a lot of ideas to improve and extend the component came into our mind, for instance regarding performance issues (e.g. caching). This Section describes the idea of tagging a little bit more in detail.

3.4.1 Tagging

While reading and reasoning about query expansion we tried to widen the research area. A tagging system for the types of services within SOCRADES could be integrated into the search component and improve the search for services too. A research group of the Telematica Instituut of Nederland did study tags in information systems. They developed and verified a TV-guide that offers recommendations to users on interesting programs according to their taste. Participants were asked to tag a number of TV programs [33].

The need of tags is justified by the fact that the users of the recommendation system use rather different vocabulary than the TV-guide originator. The original TV-guide is using different words, terminology for the descriptions. Often original descriptions are not precise or just too long and because of that reasons they are not noticed. In order to understand the originator more prior knowledge is needed that can be reasonable expected from users, cf. [33]. For that reasons the Telematica Instituut developed a TV-guide tagging systems that lets the user classify and organize the content by the means of tags. A tag can be described as a user-generated index to content. Different types of tags are recommended in [33]. With the attitude tags that reflect the participant's opinion towards a program they could improve the quality of the recommender system. And the most important outcome of [33] is the following: "We saw that participants provided a much more elaborated terminology for the genres than the genres used in the professionally added metadata." [33]

For all that reasons we could consider including a tagging concept into the Query Augmentation or the Search Engine in general. On the one hand it could serve as a recommender system with indirect or direct influence to the Query Augmentation strategies. On the other hand it could just be used to add more elaborated terminology to service descriptions and thereby improve the search results.

4

Implementation

| | |
|--|-----------|
| 4.1 SAP NetWeaver | 27 |
| 4.1.1 SAP NetWeaver Application Server | 28 |
| 4.1.2 SAP NetWeaver Development Studio..... | 29 |
| 4.1.3 Competitive products..... | 30 |
| 4.2 Java EE..... | 31 |
| 4.2.1 Java EE application model..... | 31 |
| 4.2.2 Enterprise JavaBeans | 32 |
| 4.2.3 Java Persistence API..... | 33 |
| 4.2.4 JavaServer Faces..... | 35 |
| 4.3 RESTful Web Services..... | 39 |
| 4.3.1 Yahoo! Web Services | 39 |
| 4.3.2 MediaWiki API..... | 40 |

This Chapter describes the implementation environment of this master thesis. Tools and technologies that are used for the realization of the Query Augmentation Module are explained in the following Sections. The first Section is about the SAP development tools, which are the runtime environment and the integrated development environment (IDE). Advantages and disadvantages of the SAP environment and an alternative development tool are mentioned too. Java Enterprise Edition (Java EE) and the used components are documented in the second Section. And the end of the Chapter gives a short overview about the consumed RESTful Web Services.

4.1 SAP NetWeaver

SAP NetWeaver [34] is an integrated technology platform and the technical foundation for most SAP solutions. It is a development and runtime environment and consists of a lot of components and tools. SAP NetWeaver Application Server, SAP NetWeaver Business Process Management that provides tools to model, execute, and monitor business processes, and SAP NetWeaver Portal that unifies business information and application to user specific views are some examples of components of SAP NetWeaver. The SAP NetWeaver Development Studio that offers rich functionality for developing Java applications and SAP NetWeaver Visual Composer that simplifies the generation of portal content or offers

customization by using visual user interfaces are SAP NetWeaver tools. More information about SAP NetWeaver is available on the SAP homepage [34].

Generally we use the SAP Application Server (SAP AS) as a runtime environment for the SOCRADES integration architecture and the SAP NetWeaver Development Studio (NWDS) for the development. The following Subsections describe how we made use of these NetWeaver components and how we experienced them in comparison with the GlassFish application server [35] and the NetBeans IDE [36].

4.1.1 SAP NetWeaver Application Server

“SAP NetWeaver Application Server brings together a proven infrastructure with the interoperability and flexibility of Web services technology. With this component of SAP NetWeaver, you get support for platform-independent Web services, business applications, and standards-based development.

SAP NetWeaver Application Server (SAP AS) provides an open and reliable infrastructure for deploying highly scalable Web applications and Web services.” [34]

As all application servers SAP AS offers functions and features for the development, deployment and maintenance of applications. Managed security, scalability, performance and high availability are ensured by the server. The SAP NetWeaver Application Server is especially developed for SAP solutions mainly supporting Java EE and ABAP applications. ABAP is a high level programming language developed by SAP, therefore some SAP solutions are implemented in ABAP.

We use SAP AS as a runtime environment for SOCRADES, which means that we are running Java EE applications on the server. SOCRADES uses Java Web Services Technologies, enterprise application technologies such as Enterprise JavaBeans (EJB) and Java Persistence (JPA) and Web Application Technologies such as JavaServer Faces (JSF) and JavaServer Pages (JSP). All of them are supported by the SAP Application Server that is fully compliant with the Java EE 5 standards.

Advantages

The most important advantage of using the SAP NetWeaver Application Server to run SOCRADES and therefore the Query Augmentation Module of this master thesis is that the server is a SAP solution and the foundation on which most of the SAP products run. Although SOCRADES middleware is a research project it hopefully will be integrated into the SAP products one day. For that further integration it is an advantage if the application is already running in a SAP environment. A possible integration is proved and the changes can be kept to a minimum, which finally means less work for the integration team.

Another advantage of the SAP AS is the stable environment and the high availability of the system. In comparison to the GlassFish application server [35] it seems as SAP put more weight on scalability, performance, security and availability.

Disadvantages

The fact that the SAP Application Server is stable means that it has a lot of securities and a good error handling implemented. This can also be a disadvantage. It makes the server heavy and slow. The server needs lot of resources and it is not recommended to run an instance of

the server on the developer's machine. Hence remote debugging becomes a necessity, which slows down the development process again. Altogether the deployment, first run of a newly deployed application and the debugging take a lot of time and it seems to be a big disadvantage for the SOCRADES projects. These projects are in a research and early development phase and need a lot of deployments and debugging, which would be easier and faster on a light development environment such as an application server that does not need lots of resources, that is fast and can be installed locally on the developer's machine.

The SAP implementation of Java EE 5 standards is another disadvantage of the NetWeaver Application Server. It seems that SAP's implementation of Java EE does not meet SUN's reference implementation of the standards. We especially experienced that behaviour within SAP's implementation of Java Persistence. In addition SAP's documentation of Java EE applications is rather minimal and structured unclear.

Conclusion

There is a danger of overkill for projects in that early state as the SOCRADES projects. As soon as SOCRADES is in a further state and as soon as it is more elaborated and stable this situation may change and the SAP NetWeaver Application Server will be a good, stable and secure environment. But for the first proof of concept of a research project we would prefer using a lighter application server.

4.1.2 SAP NetWeaver Development Studio

SAP NetWeaver Developer Studio (NWDS) is an integrated environment for the development of Java EE based multi-tiered business applications. It is based on the open-source IDE Eclipse [37] and provides an open and extensible development environment using Java and Web services. SAP has enhanced the standard Eclipse functionality with a comprehensive set of design, construction, and maintenance tools that cover the full software life cycle (cf. [34]).

The whole SOCRADES team is working with the same version of the NWDS, which has been defined in the specification of the project [1]. The project's code is managed in the version control system Subversion [38]. Therefore Subclipse [39], an Eclipse Team Provider plug-in providing support for Subversion within the Eclipse IDE, has to be added to the SAP default installation of the NWDS.

We do not document more details about handling and configuration of NWDS. Interested readers find details about coding and deployment with NWDS in the Appendix E Deployment with NWDS.

Advantages

Like the SAP AS the NetWeaver Developer Studio is an internally developed SAP tool thus it is supported at SAP too. That is a convenience of the NWDS. Whatever problems a developer meets, there are online resources available on [40] and in the worst case the NetWeaver developer team has to help.

Using a SAP product for the development offers the same benefits as using the SAP AS as runtime environment. The chances of a successful integration into the SAP portfolio increase as the solution already exists in a SAP environment

As NWDS is based on Eclipse it is easy to handle for everyone that is familiar with the Eclipse IDE.

Disadvantages

NWDS is a very extensive IDE and therefore it needs a lot of resources, is slow and includes a lot of features that are never needed. If one is not familiar with the IDE then it is difficult to find the proper views, perspectives, and functions. The enabling and disabling of components is very difficult as the dependencies cannot be checked reliable and one risks to get problems with essential components.

Making a clean installation of NWDS is a challenge and takes a lot of time. And after the installation we experienced problems with the environment meaning the NWDS and its components itself.

SOCRADES integration architecture is composed of several Java Enterprise Applications; most of them contain at least one EJB Module. These applications have to be deployed on two test servers that the project team owns. The individual Enterprise Applications have to be deployable for each team member from his developer machine. That is a real challenge within the NWDS environment because it reacts sensitive to changes in the environment and unfortunately each developer machine is slightly different. Thus during the development a lot of errors within the SAP components of the NWDS came up. Often we could not find a clean solution but had to work with workarounds. Several times it was necessary to create new workspaces, reinstall the IDE or even reinstall Java because of corrupt workspaces or NWDS installations.

In addition important features of the NWDS did not work or were not available:

- NWDS could only manage to connect to and deploy on one SAP AS. As we were working with two test servers, this is an inconvenience.
- Even though SAP provides a lot of plug-ins, there is no plug-in for UML modelling available for our version of the NDWS.

Conclusion

We lost several days of development with the NetWeaver Development Studio by treating general problems of the environment. As already mentioned the deployment and remote debugging of the applications on the SAP AS is slow and decelerated the implementation even more.

4.1.3 Competitive products

Because of all the disadvantages documented in the previous Subsections and the fact that the Query Augmentation Module is a pluggable component, we decided to develop the first version of the Module with NetBeans and deployed, run, and tested it on the GlassFish application server. Like this it became possible to put full concentration on the Query Augmentation Module at the early state of the development process. The implementation, deployment and debugging progressed faster and we could work more efficient.

At the time when the Query Augmentation Module and its user interface became more mature we transferred the project to NWDS and integrated it into the Application Service Catalogue.

As the development environment for SOCRADES has been defined at the beginning of the project we could not completely change to NetBeans and Glassfish.

In the advantages of the NWDS we mentioned that the SAP NetWeaver team offers support for all the NetWeaver problems. We are not sure if the NetWeaver support really is more efficient than the support teams of external, competitive products. SAP has a complex customer support system and it is always difficult to find the right contact point.

To close this Section we would say that light and fast development environments have lots of advantages in an early stage of development. In a later phase a more extensive environment maybe offers better features in terms of scalability, security and maintenance.

As we have been new to the NetWeaver Developer Studio it sometimes was really hard. Maybe the situation would have changed if at least one NWDS expert had been in our team.

4.2 Java EE

Java Platform, Enterprise Edition (Java EE) is a set of coordinated technologies and practices that enable solutions for developing, deploying, and managing multitier server-centric applications. Building on the Java Platform, Standard Edition (Java SE), Java EE adds the capabilities that provide a complete, stable, secure, and fast Java platform for enterprises. More information as well as examples are available on [28] and [41].

The following Subsections provide a short introduction to the Java technologies that have been used for the development of the Query Augmentation Module and the Query Augmentation User Interface (UI). A lot of interesting details go beyond the scope of this report. Hence, only some selected implementation details are explained.

4.2.1 Java EE application model

Java EE uses the widespread multitier model for enterprise applications. It is also known as client-server architecture that separates the presentation, the business logic and the data management. In the Java EE environment these tiers are called Client, Java EE Server and Enterprise Information System (Database) tier.

Figure 4.1 illustrates the multitier model of the Query Augmentation. The following description of the individual tiers is based on the SAP tutorial [42].

The **client tier** displays information and is responsible for interaction with the user. Dynamic HTML pages rendered from JSP pages appear as client.

Web tier components run on the Java EE server. Java EE web components are pages created using JavaServer Faces technology (see 4.2.4 JavaServer Faces). JavaServer Faces are built on servlets and JSP technology and provide a user interface component framework for web applications. Servlets are Java classes that dynamically process request and construct responses.

Business tier: Enterprise Beans (see 4.2.2 Enterprise JavaBeans) are running in the business tier on the Java EE server. An Enterprise Bean receives data from client and possible from storage, processes it, sends it to the database for storage (if necessary) and sends it back to the client program.

The bottom tier contains the **database**.

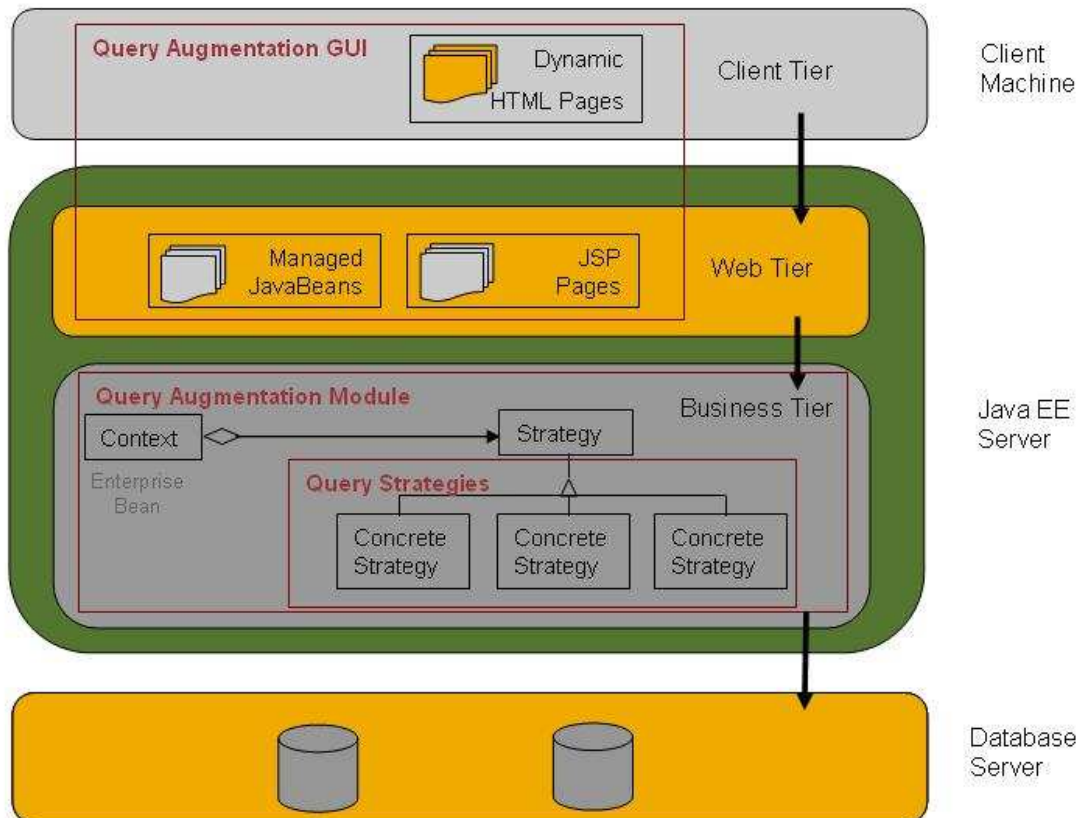


Figure 4.1 Architecture in the style of figure 1-1 of [28]

4.2.2 Enterprise JavaBeans

Enterprise JavaBeans (EJB) components are an essential part of Java EE applications. They are server-side business components that encapsulate the business logic of the application and run on the server. The EJB technology simplifies the handling of distributed applications by managing security, transaction handling, and database access (cf. [28] and [42]). Enterprise Beans are running in the EJB container, which holds a pool of instances of the Enterprise Beans and shares them between the clients.

A Java Enterprise Bean is either a **Session** (synchronous communication) or a **Message-Driven Bean** (asynchronous communication). The Query Augmentation Module only uses Session Beans that are performing tasks for a client and some of them are exposed as Web Services.

“A **Session Bean** represents a single client inside the Application Server.” [28] There are two types of Session Beans: stateless and stateful.

A **stateless Session Bean** does not maintain a conversational state with the client. State specific variables are only hold for the duration of the invocation. “The state of a stateless Session Bean should apply across all clients. Because stateless Session Beans can support multiple clients, they can offer better scalability for applications that require large numbers of clients.” [28] A particularity of stateless Session Beans is that they can be easily exposed as a Web Service by adding some annotations.

In a **stateful Session Bean**, the instance variables represent the state of a unique client-bean session. The conversational state is retained for the duration of the client-bean session (cf. [28]).

Even though the Query Augmentation Module consists of more than 20 Java classes only 4 of them are Enterprise Beans. These are listed and briefly described in Table 4.1.

| Enterprise Bean | Stateful | Stateless | WebService | Remote Interface | Local Interface | Description |
|--------------------|----------|-----------|------------|------------------|-----------------|--|
| QueryEngineBean | ✓ | | | | ✓ | This bean offers the main methods of the Query Augmentation for the UI project. The method <code>getServiceTypesByKeywords</code> is the access point and the main method of the Query Augmentation Module. |
| StrategyContext | | ✓ | ✓ | ✓ | ✓ | The StrategyContext represents the Context of the strategy pattern. It maintains a reference to the Strategy object and offers different methods (<code>getKeywords</code> , <code>getResultSet</code> , and <code>executeStrategy</code>) to use the Strategy object. |
| ConfigurationBean | | ✓ | ✓ | ✓ | ✓ | This bean is offering getters and setters for all available configuration properties. |
| StopwordFacadeBean | | ✓ | ✓ | ✓ | ✓ | This bean is managing the JPA Entity Stopword. The bean offers the methods: <code>create</code> , <code>destroy</code> and <code>isStopword</code> . |

Table 4.1 Session Beans of the Query Augmentation Module

When does it make sense to use an Enterprise Bean and when are POJOs adequate? The advantages of the Beans within the enterprise applications are the following:

- The EJB container is working with a pool of EJBs. There exist as few instances of each Bean as necessary. For POJO a new instance is created for each call.
- The Beans' lifecycle is maintained in the application server.
- The scalability of Beans is better.

For these reasons the concept of using Enterprise Beans in the Query Augmentation Module has to be reconsidered. Performance and memory usage tests could be helpful to define the classes that should run as Beans.

4.2.3 Java Persistence API

The Java Persistence API (JPA) provides a unified persistence model that means an object/relational mapping for managing relational data within Java applications. Entities,

typically entities represent tables in a relational database, are POJOs that benefit from a standardized object relational mapping using annotations.

Until now the Query Augmentation Module holds one Entity, which is the Entity Stopword for the storage and management of the stop word list. Stop words are words which are filtered out prior to processing of texts in the Query Augmentation strategies.

Code 4.1 displays the source code of the Entity Stopword. The class Stopword is a typical Java class with a lots of JPA annotations. Here are some examples. The annotation `@Id` (line 9) defines the primary key of the Entity. And the properties of the class in general are declared as columns of the database table via the annotation `@Column` (see code line 15 and 18).

```
1  @Entity
2  @Table(name = "SII_APPCAT_STOPW")
3  @NamedQuery(name="getAllStopWords",
4             query="SELECT cw FROM Stopword cw ORDER BY
5             cw.strWord")
6  public class Stopword implements Serializable {
7      private static final long serialVersionUID =
8      5310811481590016227L;
9      @Id
10     @TableGenerator(name = "idGenerator",
11                    table = "SII_TRA_ID_GEN",
12                    initialValue = 500)
13     @GeneratedValue(strategy = GenerationType.TABLE,
14                    generator = "idGenerator")
15     @Column(name = "ID")
16     private int id;
17
18     @Column(name = "STRWORD")
19     private String strWord;
20
21     public int getId() {
22         return id;
23     }
24
25     public String getStrWord() {
26         return strWord;
27     }
28
29     public void setStrWord(String strWord) {
30         this.strWord = strWord;
31     }
32 }
```

Code 4.1 Entity Stopword

Another possible use of the JPA could be the caching of results. Enriched keywords are not yet cached within the Query Augmentation Module. But the caching has to be checked as it may help improve the performance of the Query Augmentation. In the case of caching a more complex model with several entities (Query, Configuration, Result, and Keyword) and relationships between them would be necessary. Samples of such models are described in [28].

4.2.4 JavaServer Faces

JavaServer Faces (JSF) is a Java technology for the development of user interfaces. JSF can be used for creating Web user interfaces. Since Java EE 5, JSF is part of the Java EE specification. It is a server-side technology and therefore suitable for providing dynamically generated content.

In contrast to the JSP technology that is mixing presentation and programming logic JSF provides a perfect separation of components according to the Model, View and Controller (MVC) principle. Thus JavaServer Faces adhere to the Observer design pattern.

The Query Augmentation UI is built upon the JSF technology. This Subsection does describe some interesting details about the user interface but not the whole UI component. In order to understand the coherences and the functionality of the following Subsections we recommend reading Chapter 9 JavaServer Faces Technology of [28].

Backing Beans

The programming logic of a JSF application is implemented in Backing Beans. An application usually includes several Backing Beans. Each Backing Bean is coupled with the UI components in the view. These beans look like ordinary Java classes but they are JavaBeans component and therefore they offer functionalities as the Enterprise Beans. As an example it is possible to refer to Session Beans via the annotation `@EJB`. The main tasks of a Backing Bean according to [28] are:

- Handling an event fired by a UI component.
- Performing processing to determine the next page to which must be navigated.
- Validating a component's data.

Code 4.2 displays a part of the Backing/Managed Bean `QueryEngineManagedBean`. The extract shows the property `searchString` that is assigned to a form field in Code 4.4.

```
1 package com.sap.sii.appServiceCatalog;
2 ...
3 public class QueryEngineManagedBean {
4 ...
5     private String searchString = "temperature";
6     ...
7     public String getSearchString() {
8         return searchString;
9     }
10    public void setSearchString(String searchString) {
11        this.searchString = searchString;
12    }
13    ...
14 }
```

Code 4.2 Backing Bean class

The link between the Bean (programming logic) and the html web page is established in the configuration file `faces-config.xml` as displayed in Code 4.3.

```

1    <managed-bean>
2        <managed-bean-name>queryEngine</managed-bean-name>
3        <managed-bean-class>
4            com.sap.sii.appServiceCatalog.QueryEngineManagedBean
5        </managed-bean-class>
6        <managed-bean-scope>session</managed-bean-scope>
7    </managed-bean>

```

Code 4.3 Configuration of a Managed Bean

A component's value, as in Code 4.4 the value of a form's text field, can be bound to a Bean property by the tag's value attribute, which refers to the Bean property.

```

1    <h:inputText    id="keyword" label="search string"
2        value="#{queryEngine.searchString}" />

```

Code 4.4 UI component tag

The Query Augmentation UI includes four Backing Beans and four corresponding user interface pages. As suggested in the tutorials we implemented a Bean for each UI page. Backing beans mediate between the user and the persistence layer.

| Backing Bean | UI page | Description |
|-----------------------------|-------------------|--|
| ConfigurationManagedBean | configuration.jsp | This bean is offering default settings for the configuration, getters and setters to change the properties and an action for saving the changes. |
| QueryEngineManagedBean | queryEngine.jsp | This bean is offering properties (for the form fields and result data tables) and actions for the QueryEngine search and evaluation UI. |
| SearchStrategiesManagedBean | summary.jsp | Properties and actions for the calculation and representation of keyword enrichment summary are offered. |
| StopWordManagedBean | stopwords.jsp | This bean is offering properties and actions for the JPA Entity Stopword and its presentation. |

Table 4.2 Managed Beans and UI pages

Navigation model

“The JavaServer Faces navigation model makes it easy to define page navigation and to handle any additional processing needed to choose the sequence in which pages are loaded.” [28] The navigation rules for choosing the next page after a click on a button or hyperlink are defined in the `faces-config.xml` configuration file too. An outcome String from the component's action and for the next page have to be defined in the rules. The output string is used to select the right navigation rule. The outcome string is either defined in the tag attribute `action` of the UI component or within the programming logic in the Backing Bean.

The Query Augmentation UI includes very easy navigation rules (see Table 4.3). The output string "queryEngine" always lead to the UI page `queryEngine.jsp`, which is the main page of the Query Augmentation UI with the search form.

| from-outcome | to-view-id |
|-------------------|--------------------|
| queryEngine | /queryEngine.jsp |
| compareStrategies | /summary.jsp |
| stopwords | /stopwords.jsp |
| configuration | /configuration.jsp |
| get_wsdl | /wsdl.jsp |

Table 4.3 Navigation rules

Validation of a form field

The JSF framework is offering a lot of functionalities for the validation of forms. Just to show how easy the field validation became we added one example here. We extend JSF tag of the text field in Code 4.4 with the lines 4 – 6 in Code 4.5 and as a result we added two checks: First an input is required now and second the input's maximal length must not go beyond 50 characters. In addition an error message has been added that is automatically displayed to the user in case the validation fails.

```

1   <h:inputText id="keyword" label="search string"
2       value="#{queryEngine.searchString}"
3       maxLength="50"
4       required="true"
5       requiredMessage="Search term is required. Please enter
a search term."/>

```

Code 4.5 Validation

JSF HTML Tag dataTable

For the implementation of the Query Augmentation UI we used several JSF UI component tags. An overview about all existing JSF tags is given in the user guide of the JSF Toolbox on [43].

One of the most important and most complex tags we used for the implementation is the JSF HTML Tag **dataTable** (see Figure 4.2). [44] is a blog entry that describes the usage of JSF data tables very detailed and with a lot of illustrative samples. We put the focus on the following features of the data tables of the Query Augmentation UI:

- Sorting function,
- different style for alternating rows,
- the numbering of rows, and
- the paging.

All these features are displayed in Figure 4.2

numbering

The image shows a screenshot of a JSF datatable with four columns: Keyword, Wiki, Wiki Links, Wiki Backlinks, Wiki Bold and Italic Words, and Yahoo! Web Search. The rows are numbered 1 to 4. Annotations include a red arrow pointing to the row number '1' labeled 'numbering', a red arrow pointing to the 'Wiki Bold and Italic Words' column header labeled 'sorting', and a red arrow pointing to the 'last' button in the footer labeled 'paging'.

| | Keyword | Wiki | Wiki Links | Wiki Backlinks | Wiki Bold and Italic Words | Yahoo! Web Search |
|---|-----------------------------|------|------------|----------------|----------------------------|-------------------|
| 1 | temperature | 150 | 45 | 19 | 5 | 47 |
| 2 | measure | 6 | 0 | 0 | 0 | 5 |
| 3 | scales | 8 | 0 | 0 | 0 | 4 |
| 4 | weather | 0 | 2 | 8 | 0 | 4 |

first prev next last

Figure 4.2 JSF datatable

The **sorting** of a data table needs more effort of the developer and therefore it is not documented here. The chapter “Sorting datatable” of [44] is describing exactly the sorting methods we implemented. It shows how the sorting has to be included in the tag and the corresponding Backing Bean.

The assignment of different **styles for alternating rows** in data tables is done by adding the attribute `rowClasses` to the tag `h:dataTable` and assigning the CSS styles (see line 4 in Code 4.6).

For the **numbering of rows** the `h:dataTable` has to be bound to the Backing Bean by the `binding` attribute (see line 1 in Code 4.6). The row number can then be displayed in a label by using the `rowIndex` of the `dataTable` (see line 10 in Code 4.6).

The **paging** of a table is helpful if the table consists of a lot of data. First the number of rows that have to be displayed have to be defined in `h:dataTable rows` (line 5 in Code 4.6). The paging itself is managed in the footer of the data table in the tag `<f:facet name="footer">` (line 18ff. in Code 4.6). The buttons “first”, “prev”, “next” and “last” are defined and each of them is bound to a method of the Backing Bean via the attribute `action`.

```

6     <h:dataTable binding="#{summary.dataTable}"
7         value="#{summary.resultSet}"
8         var="item" headerClass="header"
9         rowClasses="row1, row2"
10        styleClass="table" rows="4">
11     <h:column>
12         <f:facet name="header">
13             <h:outputText value="#" />
14         </f:facet>
15         <h:outputText value="#{summary.dataTable.rowIndex + 1}" />
16     </h:column>
17     <h:column>
18         <f:facet name="header">
19             <h:outputText value="Keyword" />
20         </f:facet>
21         <h:outputText value="#{item.keyword}" />
22     </h:column>
23     <f:facet name="footer">
24         <h:panelGroup>
25             <h:commandButton value="first"
26                 action="#{summary.pageFirst}"
27                 disabled="#{summary.dataTable.first == 0}" />
28             <h:commandButton value="prev"
29                 action="#{summary.pagePrevious}"
30                 disabled="#{summary.dataTable.first == 0}" />
31             <h:commandButton value="next" action="#{summary.pageNext}"
32                 disabled="#{summary.dataTable.first +
summary.dataTable.rows >= summary.dataTable.rowCount}" />
33             <h:commandButton value="last" action="#{summary.pageLast}"
34                 disabled="#{summary.dataTable.first +
summary.dataTable.rows >= summary.dataTable.rowCount}" />
35         </h:panelGroup>
36     </f:facet>
37 </h:dataTable>

```

Code 4.6 JSF datatable

This example illustrates the simplicity JSF offers in order to display complex data structures with useful features such as sorting and paging.

4.3 RESTful Web Services

The consumption of web resources is one of the main functions of the different strategies that are used for the Query Augmentation. Seven of the eight implemented strategies are using the same approach. They are consuming RESTful web services. REST stands for Representational State Transfer and basically means that each URI represents an object. The content of the object can be obtained by using an HTTP GET request.

4.3.1 Yahoo! Web Services

The first group of RESTful web services we are consuming are the Yahoo! Web Services. “Yahoo! Search Web Services operate via HTTP requests just like your web browser. Requests are formed by starting with a service entry point URL (such as

<http://search.yahooapis.com/WebSearchService/V1/webSearchfor> web search queries) and adding query arguments to specify the search results desired (such as `?query=Madonna`). The results returned by the service are in XML which varies per service.” [45]

In particular we used the following two Yahoo! Web Services:

- My Web 2.0 Web Services: Web Search [7]
- My Web 2.0 Web Services: Related Tags [8]

Running examples as well as lots of code snippets are available on Yahoo!’s developer network [46].

4.3.2 MediaWiki API

The second group of web services was the ones offered by the MediaWiki API [6] for querying Wikipedia.

The basic approach is the same as already described in the paragraph above. A HTTP request is launched with an URL that contains variable query arguments and then returns an xml document with the desired answer.

The following three basic queries are used within the Query Augmentation strategies:

3. Latest revision of the Wikipedia article “Radio-frequency identification”:

```
http://en.wikipedia.org/w/api.php?format=xml&action=query&prop=visions&titles=RFID&rvprop=content&redirects
```

4. Links within the latest revision of the Wikipedia article “Radio-frequency identification”:

```
http://en.wikipedia.org/w/api.php?format=xml&action=query&prop=links&pllimit=500&titles=RFID&redirects&plnamespace=0
```

5. Backlinks to the Wikipedia article “Radio-frequency identification

```
http://en.wikipedia.org/w/api.php?format=xml&action=query&list=backlinks&bltitle=Radio-frequency\_identification&bllimit=500&blnamespace=0
```

One of the advantages of RESTful web services is that the result is readable for machines and humans. Therefore the queries above can be executed in a normal web browser and the result will be returned and displayed in xml structure.

5

User manual

| | |
|---|-----------|
| 5.1 Search Service Instances | 42 |
| 5.2 User Preferences | 43 |
| 5.3 Stop Words | 43 |
| 5.4 Configuration..... | 44 |
| 5.5 Query Augmentation Summary | 47 |
| 5.6 Screencasts..... | 48 |

For the visualization of the Query Augmentation we developed a web user interface to the Application Service Catalogue focusing on the Query Augmentation Module. The web project is part of the enterprise project of the Application Service Catalogue and is implemented with the JavaServer Faces technology [47].

The main objectives of the user interface are offering functionalities and information for the testing and evaluation of the Query Augmentation Module as well as the visualization of the augmentation and search. The Query Augmentation UI is not the administrative interface of the Application Service Catalogue but it exists in parallel. In contrast to the Query Augmentation UI the SOCRADES administration application does not show any details about the Query Augmentation Module. SOCRADES users usually do not have to know how the search of services is working internally.

This Chapter describes the use of the Query Augmentation Module and UI. The deployment of the Query Augmentation Module and its UI on a SAP NetWeaver Application Server is documented in Appendix E Deployment with NWDS.

The Query Augmentation UI is available on SAP NetWeaver Application Servers that have SOCRADES middleware installed. As it is a web frontend, it can be reached by using a web browser to access an URL like this one:

http://SERVERNAME:50000/SII_application_service_catalog_ui

At the top of each web page of the Query Augmentation UI the menu bar (Figure 5.1) is displayed. The five menu points (Search Service Instances, User Preferences, Stop Words, Configuration, and Query Augmentation Summary) and the functionalities behind them are explained in detail in the next Subsections.

Choosing a menu point or switching between menus is done by clicking on the links in the menu bar.



Figure 5.1 UI - Menu

5.1 Search Service Instances

The main page of the Query Augmentation UI is the “Search Service Instances” web page (Figure 5.2), which is the starting page of the application too.

“Search Service Instances” offers the user the possibility to find running instances of services on smart devices. The user enters a search term into the Search Form, launches the search and gets back an ordered list of matching types of services in the Search Result.

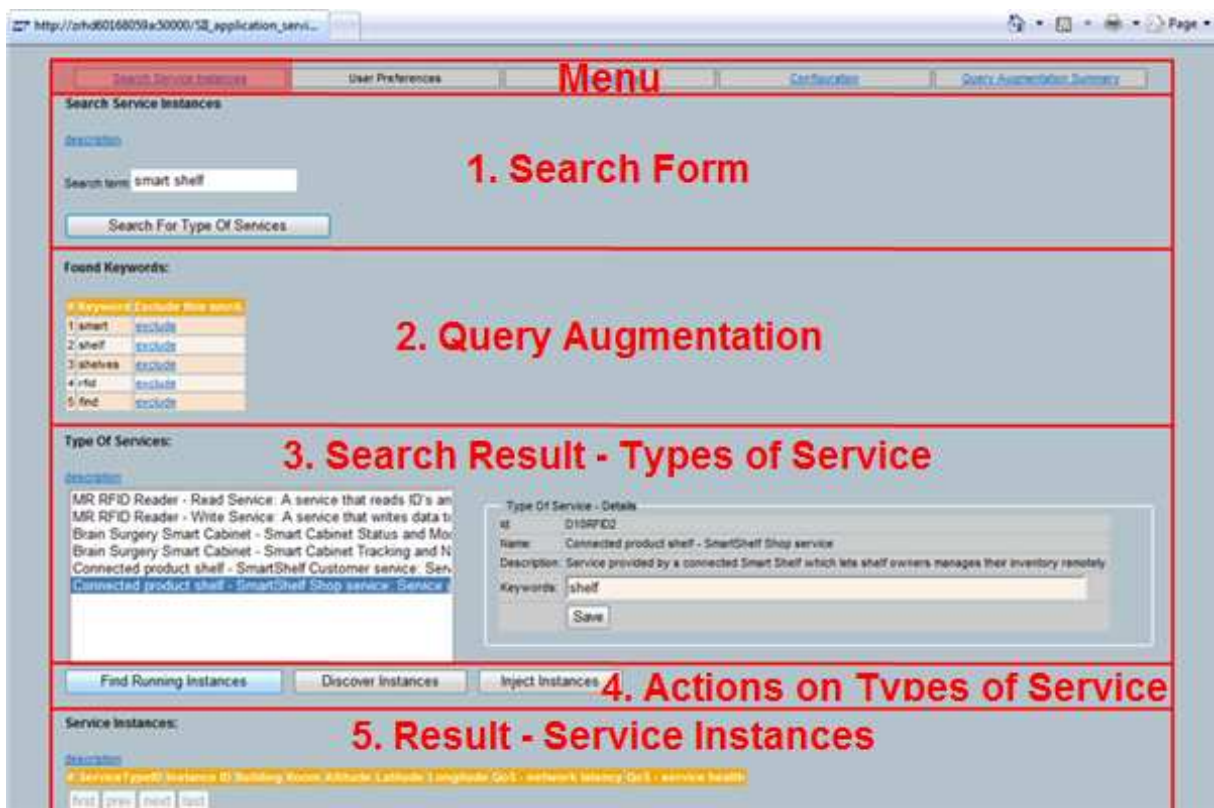


Figure 5.2 UI - Search for Service Instances

The page (Figure 5.2) is divided into Sections, which represent the steps of the search process. We consider the following steps:

- 1. Search Form:** Enter one search term into the input field ‘search term’. A search term usually is one single word, e.g. temperature, but it is also possible to insert one composed of two or three words, e.g. smart shelf. Then click the button ‘Search For Types Of Service’ to start the query enrichment and after that the search for types of services automatically.
- 2. Query Augmentation:** The Table ‘Found Keywords’ shows a list of enriched keywords. For the search of types of service not the initial search term but the list of keywords is used for the execution of the query.

The list of keywords may include common words that should be ignored. Such words can be excluded by clicking on the link next to is in the right column. After the exclusion of the word the search has to be launched again. The banned word is added to the list of stop words (see 5.3 Stop Words) and is ignored this time.

3. **Search Result – Types of Service:** The search returns a list of types of services. If only selecting one type of service of the list the details of the selected type are displayed in the right part of the Section search result.

Find the desired types of service in the result and select one or several of them for further processing.

4. **Actions on Types of Service:** The user has now three possibilities in order to get the desired service instance.
 1. **Find Running Instances:** search for running service instances within SOCRADES middleware. This should always be the first action done after one found the types of service. If no service instances are returned then one can go further to the next actions.
 2. **Discover Instances:** actively try to discover services instances in the network.
 3. **Inject Instances:** inject a type of service on a smart device that has the capability to offer the selected type of service.

All these actions may end in a timeout and do not return any result.

5. **Result – Service Instances:** The Table ‘Service Instances’ lists service instances on smart devices found for the selected types of service. Properties such as type of service, building, room, altitude, latitude, longitude, network latency, and service health are displayed directly in the result set. The service instances are ordered by these properties. So the better a service seems to fit the higher it is listed.

The result set also includes the link to the WSDL file of the service instance. The WSDL is necessary for the further use of the service, e.g. to build a web service client or to integrate the service in a business application.

5.2 User Preferences

The menu point ‘User Preferences’ is destined for user specific settings such as the current position, meaning location of the user. However a UI for the user preferences is not implemented yet.

The user preferences belong to the subcomponent Query Context, which is another query optimization component besides the Query Augmentation Module. At the design phase it was not predictable if the Query Context could be integrated completely into the UI or not. For that reason the menu point is already there.

5.3 Stop Words

Stop words is the name given to words which are filtered out prior to, or after, processing of natural language data (text). Common word such as ‘the’, ‘of’, ‘on’, and ‘a’ are stop words. They are ignored when used in a query (cf.[48]).

The Query Augmentation Module is eliminating stop words from the enriched keywords. In this case there exists one global stop word list, which is stored in the projects database. This list is not language specific. Stop words can be edited by the user of the Query Augmentation UI. Words can be removed from in the view 'Stop Words' (see Figure 5.3). Adding a word to the list can be done either at the time of the search for service instances on smart devices (see 6.1.1 Search Service Instances) or from the 'Query Augmentation Summary' (see 5.5 Query Augmentation Summary).

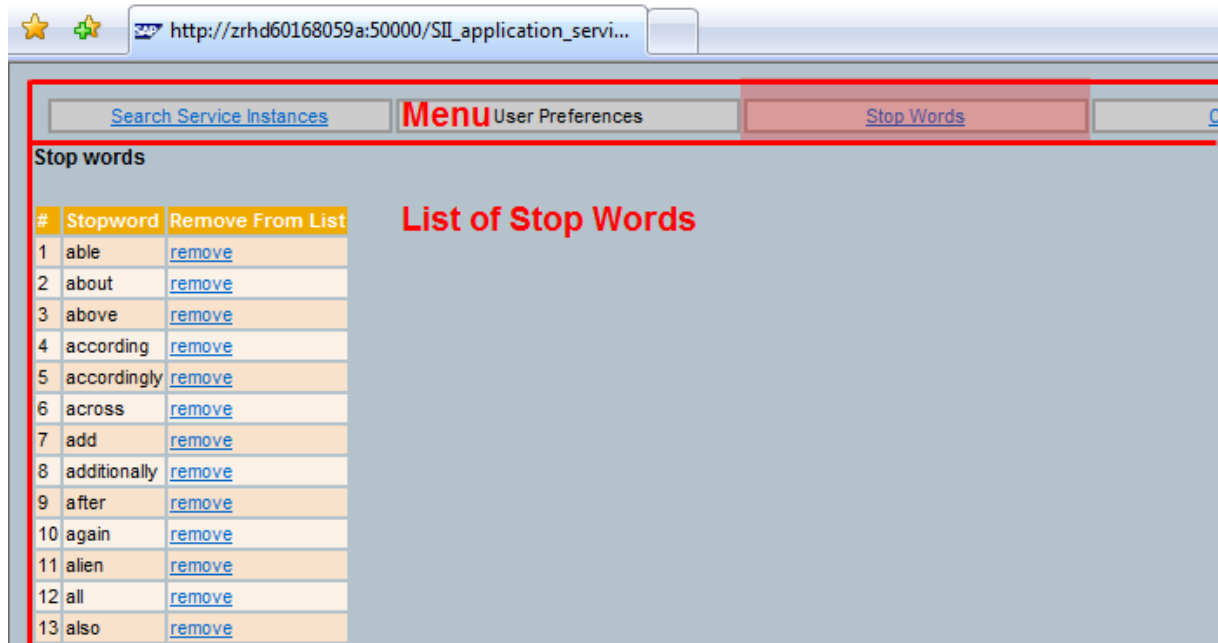


Figure 5.3 UI - Stop Words

5.4 Configuration

The flexible design of the Query Augmentation Module results in some variable configuration parameter. All these variables are included into the form of the configuration page and can be configured there.

The ability to switch on/off the Query Augmentation and to define the strategies used for the augmentation are the most important properties (see the yellow marking in Figure 5.4). For that reason these configuration parameters are explained after the next Figure. All the other configuration parameters are explained in Table 5.1 and details about the individual augmentation strategies are documented in the Chapter 3 Architecture.

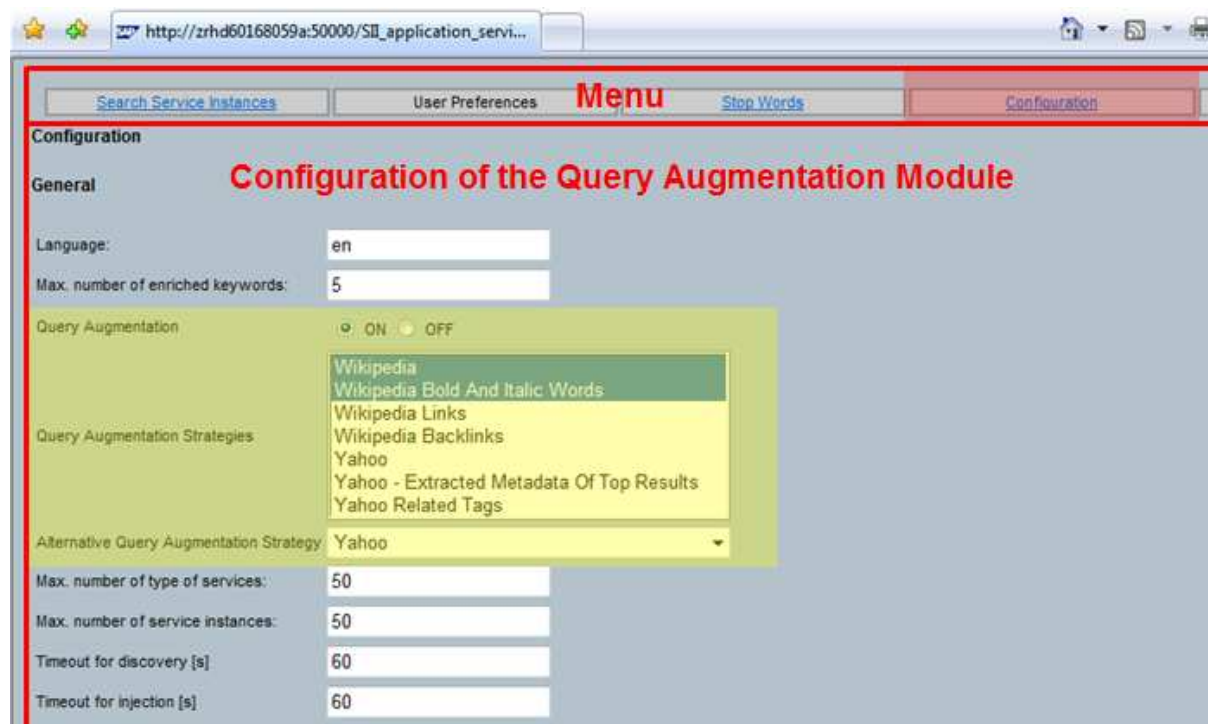


Figure 5.4 UI - Configuration

Query Augmentation

Default value: ON

This property allows the user to switch on or off the Query Augmentation. We recommend turning on Query Augmentation.

Query Augmentation Strategies

Default value: Wikipedia

This configuration parameter is enabling the combination of strategies. At present the Query Augmentation Module is offering seven different strategies that can be used for the keyword enrichment. The strategies can be used individual or in any possible combination.

Combining strategies means that the query enrichment will be executed for all the selected strategies and the final result will be calculated on the basis of the result of the individual strategies.

Alternative Query Augmentation Strategy

Default value: Yahoo

The Alternative Query Augmentation Strategy is enabling the chaining of strategies. Only if the defined strategy or strategies of the previous parameter “Query Augmentation Strategies” do not return a result and if defined, the alternative strategy will be executed and hopefully return a result.

As alternative augmentation strategy we recommend to define Yahoo (Yahoo Web Search). The Yahoo strategy is based on the common Yahoo Web Search and therefore the probability of getting a result is elevated.

Changes of the configuration always have to be saved by clicking on the save button on the bottom of the page. A confirmation message will appear at the top of the form if the changes have been saved successfully.

All configuration parameters explained:

| Parameter | Default value | Description |
|---|--|---|
| General | General configuration parameters. | |
| Language | en | Defining the language in which the search for types of service will be processed. |
| Max. number of enriched keywords | 5 | The number of search terms that will be used for the search. We recommend processing the search with an enriched query of 5 keywords. |
| Query Augmentation | ON | <i>See description in the text right after the screenshot of the configuration page (Figure 5.4).</i> |
| Query Augmentation Strategies | Wikipedia | <i>See description in the text right after the screenshot of the configuration page (Figure 5.4).</i> |
| Alternative Query Augmentation Strategy | Yahoo | <i>See description in the text right after the screenshot of the configuration page (Figure 5.4).</i> |
| Max. number of types of service | 50 | The maximal number of types of service that will be returned and displayed. |
| Max. number of service instances | 50 | The maximal number of service instances that will be returned and displayed. |
| Timeout for discovery [s] | 60 | The timeout for the active discovery of running service instances on smart devices. |
| Timeout for injection [s] | 60 | The timeout for the injection of service on smart devices. |
| Wikipedia | Parameters concerning the Wikipedia strategies. Special words such as links, backlinks, bold and italic word of a Wikipedia article can be regarded as more important and according to this they can be weighted as more important within the Query Augmentation. | |
| Weight of links | 2 | The augmentation strategy Wikipedia Links is considering links of Wikipedia articles and weights them according to the configuration. |
| Weight of backlinks | 2 | The augmentation strategy Wikipedia Backlinks is considering backlinks to Wikipedia articles and weights them according to the configuration. |
| Weight of bold and italic words | 2 | The augmentation strategy Wikipedia Bold And Italic Words is considering word written bold or italic within Wikipedia articles and weights them according to the configuration. |
| Yahoo! Web Search | Parameters concerning the Yahoo strategies. | |
| Number of Yahoo Web Search Results | 25 | Defining the number of search results that are considered for the Yahoo strategy. |

| Parameter | Default value | Description |
|---|--|--|
| Number of Detailed Yahoo Web Search Results | 10 | Defining the number of search results considered for the strategy Yahoo – Extracted Metadata Of Top Results. |
| Environment – Proxy Server | Proxy settings of the SAP Web AS Enabling the connection to the internet via proxy server and set the following Java properties. (cf. [49]) | |
| proxySet | true | |
| proxyHost | proxy | |
| proxyPort | 8080 | |

Table 5.1 Configuration parameters

5.5 Query Augmentation Summary

The Query Augmentation Summary is representing the detailed composition of the Query Augmentation result.

Figure 5.5 and Figure 5.6 show the page, which has the Menu at the top of the page as usual, followed by a Section Form and a detailed Table called Query Augmentation Summary.

Form

Entering the same search term into field Term as for the search of types of service will execute the same Query Augmentation as the one included in the search for types.

The summary does return the number of keywords defined in the field Limit. It makes sense to define a limit between 100 and 500. The bigger the limit the better one can see and understand the structure of the result.

Clicking the button Compare the query augmentation strategies will execute the Query Augmentation as defined in the Configuration for the entered search term. The returned result is displayed in the Table Query Augmentation Summary that is described in the next Section.

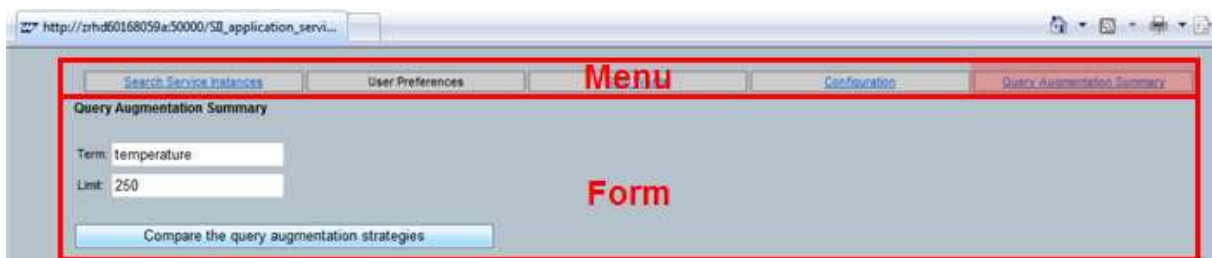


Figure 5.5 UI - Query Augmentation Summary – part I

Query Augmentation Summary

To understand the result and its composition it is not necessary to know any internal details about the augmentation strategies. The only thing that has to be understood is that the result of each strategy is a list of keywords. Each keyword holds a weight which defines the rank of the keyword in the list. The list of keyword is sorted by this weights meaning that the largest number is at the top of the list.

If a combination of query enrichment strategies is configured then the total weight is calculated by adding the results of the individual strategies together.

We now look at the Table of the Query Augmentation Summary in Figure 5.6. The first and second column (rank and keyword) represent the result of the Query Augmentation as it is presented on the page Search Service Instances (Subsection 6.1.1). In addition the Query Augmentation Summary has one column for each individual augmentation strategy and the column TOTAL with the overall result.

In order to analyze the result and to compare strategies the Table can be sorted by each result column. By default it is ranked in descending order of weights of the TOTAL.

When changing the sorting we can observe how the order of the keyword changes. Therefore this summary can help us to find the best combination of augmentation strategies with the best keywords at the top of the Table.

Attention: comparing the query augmentation strategies may take several minutes.

| Keyword | Wiki | Wiki Links | Wiki Backlinks | Wiki Bold and Italic Words | Yahoo! Web Search | Yahoo! Meta Data Keywords | Yahoo! Related Tags | TOTAL |
|---------|----------------|------------|----------------|----------------------------|-------------------|---------------------------|---------------------|-------|
| 1 | temperatura | 150 | 45 | 19 | 5 | 44 | 41 | 305 |
| 2 | heat | 32 | 14 | 12 | 2 | 3 | 2 | 65 |
| 3 | scale | 31 | 18 | 4 | 0 | 3 | 1 | 57 |
| 4 | energy | 32 | 12 | 10 | 0 | 2 | 0 | 56 |
| 5 | system | 36 | 2 | 12 | 0 | 1 | 2 | 53 |
| 6 | gas | 27 | 10 | 6 | 2 | 2 | 0 | 47 |
| 7 | thermal | 23 | 6 | 4 | 0 | 0 | 9 | 42 |
| 8 | thermometer | 14 | 2 | 4 | 2 | 2 | 13 | 37 |
| 9 | pressure | 8 | 10 | 14 | 0 | 2 | 0 | 34 |
| 10 | point | 13 | 10 | 10 | 0 | 0 | 0 | 33 |
| 11 | law | 18 | 6 | 8 | 0 | 0 | 0 | 32 |
| 12 | zero | 24 | 4 | 2 | 0 | 0 | 0 | 31 |
| 13 | thermodynamics | 14 | 10 | 6 | 0 | 0 | 0 | 31 |
| 14 | physics | 7 | 10 | 8 | 0 | 2 | 1 | 28 |
| 15 | absolute | 18 | 2 | 4 | 0 | 2 | 1 | 27 |
| 16 | equilibrium | 17 | 4 | 4 | 0 | 0 | 0 | 25 |
| 17 | measurement | 6 | 6 | 2 | 0 | 3 | 6 | 25 |
| 18 | matter | 9 | 6 | 8 | 0 | 2 | 0 | 25 |
| 19 | celsius | 12 | 4 | 2 | 0 | 1 | 2 | 25 |
| 20 | radiation | 12 | 8 | 2 | 0 | 0 | 2 | 24 |
| 21 | kinetic | 14 | 6 | 2 | 0 | 1 | 0 | 23 |
| 22 | weather | 0 | 2 | 8 | 0 | 4 | 3 | 23 |
| 23 | thermodynamic | 11 | 4 | 6 | 0 | 1 | 0 | 22 |
| 24 | fahrenheit | 6 | 6 | 4 | 0 | 1 | 2 | 21 |
| 25 | definition | 14 | 0 | 2 | 0 | 2 | 0 | 20 |

first prev next last

SOCRADES - Service Application Catalog 10/30/2008 17:15:24

Figure 5.6 UI - Query Augmentation Summary – part II

5.6 Screencasts

For demonstration purposes we created two screen casts, which are presenting the Query Augmentation UI in action. The screen casts are published as .pdf files on the projects website [27].

Demo I – SmartShelf.pdf explains the search form of the UI in general and shows the search and integration of a smart device. The smart device we are looking for is a connected product shelf that has to be integrated into a new business application at development time.

Demo II – Benefits of the query augmentation.pdf demonstrates the advantages of the Query Augmentation. This time we are looking for a water meter, first without the use of the Query Augmentation Module and then with query enrichment.

6

Experimental Evaluation

| | |
|---|-----------|
| 6.1 Methodology | 50 |
| 6.1.1 Data collection | 50 |
| 6.1.2 Test data | 52 |
| 6.1.3 Scenarios | 53 |
| 6.2 Results | 53 |
| 6.2.1 Best configuration | 54 |
| 6.2.2 Wikipedia strategy | 57 |
| 6.2.3 Yahoo Related Tags strategy | 59 |
| 6.3 Discussion | 60 |
| 6.3.1 Future evaluation | 61 |

We have designed the Query Augmentation Module on the base of the existing specification of the SOCRADES project [1], implemented it with a set of query augmentation strategies (see Chapter 3, Section 3.1 Query strategies), integrated it into the SOCRADES Application Service Catalogue and finally added a web user interface called Query Augmentation UI or just UI (Chapter 5). The UI has been designed and implemented for the testing of the Application Service Catalogue, especially for the evaluation of the Query Augmentation Module. The evaluation and the answers of the research questions are documented in this Chapter.

The approach of evaluation (6.1 Methodology) is described first, then the results of the evaluation (6.2 Results), which serve as a basis for answering the research questions, are presented. The experimental results show the effect and applicability of the Query Augmentation within SOCRADES Application Service Catalogue. Finally the interpretation of the results and inputs for more evaluation is provided in the last Section (6.3 Discussion).

The mentioned research questions are listed in Chapter 1.4 Research questions. A summary of the questions is provided here:

1. Does Query Augmentation increase the result quality of the search?
2. Do strategies based on human generated texts lead to better results than others?
3. Which is the optimal strategy?

6.1 Methodology

How did we process our tests? How did we get the test data? These two questions are mainly answered in the following three Subsections.

6.1.1 Data collection

The main task of the evaluation is to find for service instances running on smart devices that are integrated into the SOCRADES middleware. Real smart devices are not yet available at SAP Research and therefore we have to work with virtual smart devices. We first have to generate a set of test data meaning a set of service instances running on virtual smart devices.

The generation of the test data was done by processing the following seven steps:

1. Defining search terms

At the very beginning we defined 10 search terms that are associated with potential smart devices. Search terms meaning keywords, which represent the initial query of the search for service instances on smart devices.

The defined search terms are 1. RFID, 2. temperature, 3. sensor, 4. conveyor belt, 5. access control, 6. gripper, 7. vibration, 8. location sensor, 9. smart meter, and 10. webcam.

That set is used again and again in the evaluation.

2. Defining smart devices

According to the search terms we defined 19 smart devices, about two devices for each search term. In this case smart device means a device that has the potential of offering access to functionalities through web services. Maybe it does not have this quality yet. E.g. for the search term RFID a RFID Smart Shelf [50] has been defined as potential smart devices, or for the term smart meter a water meter [51]. The SunSPOT [52] is a special device with a large application spectrum and for that reason this device is associated with several search terms and not with one term only.

A detailed list of all the virtual smart devices used in this evaluation is added to the appendix; see B Smart Devices.

3. Design user study

The next step was to describe potential services offered by the smart devices. This part of the data collection we did within a user study among neutral persons that did not know the SOCRADES project before. We generated a questionnaire for each of the smart devices whose content is a short project description and a form to describe web services. These descriptions will be used to generate the actual test data later in the process.

In total 19 questionnaires have been created. An example is added to the appendix, see C Questionnaire for User Study.

4. Handing out the questionnaires

We handed out the questionnaires to neutral people. Each smart device has been sent to one person except from the SunSPOT device, which has been sent to several persons.

In total 21 people got the user study and had two weeks to return it. Being familiar with web services was the only requirement the people had to meet.

5. Register the returned questionnaires

After the expiration of a deadline we got 17 answers and two of them concerned the SunSPOT device. As a result we had 16 devices with one or two services.

From the questionnaires we extracted all information needed and generated the machine readable service description file, the WSDL file, for each service. That resulted 30 WSDL files (i.e. 30 types of service) out of the questionnaires. A sample of a WSDL file can be consulted in the appendix D WSDL file.

Here is an example of services a smart device could offer. The RFID Smart Shelf offers: SmartShelf Customer service offering operations like `hasProduct(description)` and SmartShelf Shop service offering operations like `getProductCount(description)`.

6. Integrate the result of the user study

The SOCRADES integration architecture is not yet ready to detect and include running services automatically. So we did not build running services from the WSDL files but we registered the described services as types of service in the SOCRADES Service Implementation Repository. With the SOCRADES Service Monitor, another component of the integration architecture, we then generated a bunch of services instances for the registered types of service.

In total we registered 30 types of service for the 16 smart devices in the Service Implementation Repository. Most of the smart devices offer 2 different types of service. Out of these types of services we auto-generated 1000 running service instances within the Service Monitor. The Service Monitor is managing the smart devices and the service instances. We do not know on how many virtual smart devices our test service instances are running but the Service Monitor has control over them.

The SOCRADES middleware is now ready to be tested via the Query Augmentation UI of the Application Service Catalogue. We will be able to find the types of services as well as the service instances.

7. Define additional search terms

In the first step we defined the search terms. With these terms we check if the right types of service will be found. Of course there is not only one search term that should lead to a service instance. The users may have different background knowledge and search terms, but finally they are looking for the same smart device and its services. So we should have some additional search terms for 16 smart devices.

We presented photos of the smart devices to another four neutral persons and asked them for the search terms they would use in order to search for services on the devices. Like this we got a larger collection of search terms, which we later used for the evaluation of the augmentation strategies. The complete list of search terms for each smart device can be found in appendix B Smart Devices.

6.1.2 Test data

This Section contains some explanations how the search terms, devices, types of service and service instances have been used in the evaluation. We structured the data in order to be able to summarize intermediate results and finally being able to judging the different configuration of the Query Augmentation Module.

The basic schema, illustrated in Table 6.1, is used for most of the evaluation and is based on the search terms defined at the very beginning of the data collection. For several reasons we had to exclude the search term “location sensor” so nine search terms remained.

Table 6.1 explained:

The header contains the search terms and three columns for the analysis at the end.

The row Expected represents the number of types of service registered in the test environment for the according search term, e.g. there are 6 services in the system that should be returned if launching the search with the query RFID. Using nine search terms and adding up the number of expected types of service results in a total of 36 services or 100%.

False Pos. (false positives) is the row which contains the number of types of service that have been found with a certain search term, but were not expected. Each request is searching for matching types of service in the test data of 30 types. Thus there is always a potential to find 30 types of services and not only the expected ones. The optimal number of false positives is 0 that is 0%.

The last row represents the maximal number of false positives possible in each search. As already mentioned we have 30 types of service registered. If we expect 6 types within the result set, there are 24 types of service that could be returned as false positives. The sum of all the false positives is 234, which is 87% of all the types ($234 + 36 = 270 \hat{=} 100\%$). 87% means that within our test system the false positives never exceed this mark and the more false positives the more noise exists in the result set.

| Search term | RFID | temperature | sensor | conveyor belt | access control | gripper | vibration | smart meter | webcam | TOTAL | Found | False Pos. |
|------------------------|------|-------------|--------|---------------|----------------|---------|-----------|-------------|--------|-------|-------------|------------|
| Expected | 6 | 6 | 6 | 2 | 2 | 3 | 6 | 4 | 1 | 36 | 100% | |
| False Pos. | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | 0% |
| Max. False Pos. | 24 | 24 | 24 | 28 | 28 | 27 | 24 | 26 | 29 | 234 | | 87% |

Table 6.1 Basic structure of test data

This schema always requires the result of nine search processes and it can be used with any configuration of the Query Augmentation Module.

6.1.3 Scenarios

This Section does roughly describe the main approach of the Query Augmentation evaluation. All the details necessary to understand the results are documented within the results (6.2 Results).

Procedure

The general and repetitive scenario of the practical evaluation consists of the following steps:

- Define the search strategies (no strategy, individual strategy, combination, chain, or combination and chain of strategies).
- Define the number of search terms (number of enriched keywords) for the search of types of service.
- Perform nine search requests. Nine searches mean one search request for each predefined search term.
- Sum up the individual results to one result which is then used for the comparison with other configurations.

Configurations

Before starting the evaluation based on this general scenario, we have to set limits to the configuration. Within the scope of this master thesis not all possible combinations of strategies can be tested. Therefore we defined a selection of configurations that have to be considered. The following limitations have been defined:

In order to find the optimal number of search terms, the tests have to be performed with the following **number of search terms**: 1, 5, 10, 20, and 50.

For the purpose of finding the best strategy, or to put it better the best configuration of the strategies, the following **Augmentation strategies** have to be evaluated:

- No strategy (meaning Query Augmentation is turned off, meaning one search term),
- each strategy individually,
- combination of all Wikipedia strategies,
- combination of the strategies Yahoo and Yahoo – Extracted Metadata Of Top Results,
- and combination of Wikipedia and Yahoo.

Performing the above documented procedure with the above mentioned configurations now, will allow us to answer if Query Augmentation is useful and if strategies based on human generated texts are better. Furthermore we should be able to find the optimal configuration of the Query Augmentation Module based on the tests executed.

6.2 Results

First the benefit of the Query Augmentation Module in general is shown and the optimal configuration is described. Then we take a closer look into two concrete strategies, Wikipedia and Yahoo Related Tags.

One preliminary remark before we are presenting the results. We excluded the strategy DBpedia from the evaluation. This strategy has been implemented in order to demonstrate how an ontology can be accessed over a SPARQL [22] query endpoint. In general it is just another possibility to access the content of Wikipedia. For that reasons the strategy DBpedia is neither integrated into the Query Augmentation UI nor is it evaluated.

6.2.1 Best configuration

Defining the best configuration of the Query Augmentation Module contains mainly two parameters. First we try to define the optimal number of enriched keywords for the query. Then we define the most favourable configuration of the augmentation strategies.

Number of search terms

This part of the evaluation does not only define the optimal number of search terms, it also proves that using Query Augmentation does improve the result of the search for types of service.

In order to find the best number of search terms we compared the results of search requests, which made use of 1, 5, 10, 20, and 50 enriched keywords. One enriched keyword means that the Query Augmentation has been switched off and the query only contains the initial search term.

The result is illustrated in Figure 6.1. The data series Wikipedia and Yahoo Web Search represent the success of the Query Augmentation meaning the percentage of found services. The closer a data point to 100% is the better.

For better tractability the exact configuration of the data series:

Wikipedia: Combination of the strategies “Wikipedia”, “Wikipedia Bold and Italic Words”, “Wikipedia Links”, and “Wikipedia Backlinks”.

Yahoo Web Search: Combination of the strategies “Yahoo” based on the top 25 web search results and “Yahoo - Extracted Metadata Of Top Results” collected from the top 10 web search results.

Both data series (Wikipedia and Yahoo Web Search) converge to 100% if increasing the number of search terms. That result was expected. It is predictable that the success is 100% when using a dictionary in the query.

But it would be wrong to say that using 50 enriched search terms is optimal for the Query Augmentation Module. We have to look at the noise that is created by these requests too. The noises in our case are the false positives. Each request may return types of service that have not been expected but also matched to the query. These additional types of service in the result are called the false positives and they have to be kept as few as possible. The false positives are also represented in Figure 6.1. The data series Max. False Positives shows the maximal of false positives that are possible within our test system and the data series Wikipedia False Positives and Yahoo False Positives display an increase of false positives with the rise of search terms.

Therefore finding the optimal number of search terms for the Query Augmentation is a trade-off between the success of the Query Augmentation (data series Wikipedia and Yahoo Web

Search) and the false positives (data series Wikipedia False Positives and Yahoo False Positives), which have to be kept low.

The Query Augmentation with 5 search terms is optimal. The increase of the marginal success rate from 1 to 5 search terms is the highest in the graph and shows a clear benefit of the use of Query Augmentation. At the same time the results contains less then 20% of false positives.

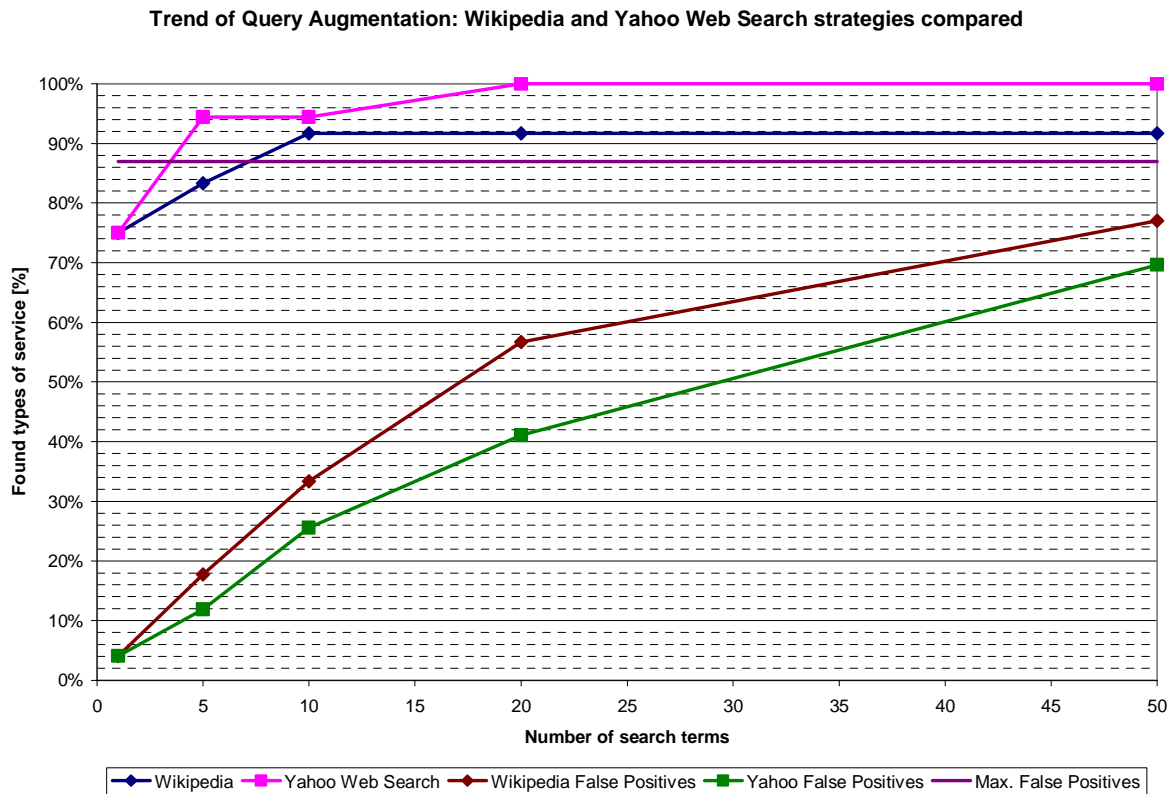


Figure 6.1 Trend of Query Augmentation

Table 6.2 represents the data source of the Figure above.

| Search term | RFID | temperature | sensor | conveyor belt | access control | gripper | vibration | smart meter | webcam | TOTAL | Found | False Pos. |
|------------------------------------|------|-------------|--------|---------------|----------------|---------|-----------|-------------|--------|-------|-------|------------|
| Wikipedia – SERVICES FOUND | | | | | | | | | | | | |
| No augmentation | 6 | 6 | 6 | 0 | 1 | 3 | 2 | 2 | 1 | 27 | 75% | |
| 5 keywords | 6 | 6 | 6 | 0 | 2 | 3 | 2 | 4 | 1 | 30 | 83% | |
| 10 keywords | 6 | 6 | 6 | 2 | 2 | 3 | 3 | 4 | 1 | 33 | 92% | |
| 20 keywords | 6 | 6 | 6 | 2 | 2 | 3 | 3 | 4 | 1 | 33 | 92% | |
| 50 keywords | 6 | 6 | 6 | 2 | 2 | 3 | 3 | 4 | 1 | 33 | 92% | |
| Wikipedia - FALSE POSITIVES | | | | | | | | | | | | |
| No augmentation | 0 | 2 | 0 | 0 | 5 | 0 | 0 | 4 | 0 | 11 | | 4% |
| 5 keywords | 3 | 9 | 7 | 5 | 8 | 0 | 7 | 4 | 5 | 48 | | 18% |

| Search term | RFID | temperature | sensor | conveyor belt | access control | gripper | vibration | smart meter | webcam | TOTAL | Found | False Pos. |
|--------------------------------|------|-------------|--------|---------------|----------------|---------|-----------|-------------|--------|------------|-------------|------------|
| 10 keywords | 9 | 9 | 13 | 28 | 12 | 1 | 9 | 4 | 5 | 90 | | 33% |
| 20 keywords | 24 | 9 | 19 | 28 | 13 | 17 | 11 | 15 | 17 | 153 | | 57% |
| 50 keywords | 24 | 24 | 24 | 28 | 28 | 27 | 11 | 23 | 19 | 208 | | 77% |
| Yahoo – SERVICES FOUND | | | | | | | | | | | | |
| No augmentation | 6 | 6 | 6 | 0 | 1 | 3 | 2 | 2 | 1 | 27 | 75% | |
| 5 keywords | 6 | 6 | 6 | 0 | 2 | 3 | 6 | 4 | 1 | 34 | 94% | |
| 10 keywords | 6 | 6 | 6 | 0 | 2 | 3 | 6 | 4 | 1 | 34 | 94% | |
| 20 keywords | 6 | 6 | 6 | 2 | 2 | 3 | 6 | 4 | 1 | 36 | 100% | |
| 50 keywords | 6 | 6 | 6 | 2 | 2 | 3 | 6 | 4 | 1 | 36 | 100% | |
| Yahoo - FALSE POSITIVES | | | | | | | | | | | | |
| No augmentation | 0 | 2 | 0 | 0 | 5 | 0 | 0 | 4 | 0 | 11 | | 4% |
| 5 keywords | 5 | 2 | 4 | 0 | 11 | 0 | 2 | 4 | 4 | 32 | | 12% |
| 10 keywords | 5 | 5 | 12 | 6 | 13 | 8 | 10 | 6 | 4 | 69 | | 26% |
| 20 keywords | 9 | 10 | 13 | 11 | 19 | 8 | 11 | 26 | 4 | 111 | | 41% |
| 50 keywords | 24 | 24 | 21 | 15 | 28 | 14 | 24 | 26 | 12 | 188 | | 70% |

Table 6.2 Optimal number of search terms

Augmentation strategies

After the definition of the best number of search terms for the query we expanded the evaluation to other configurations of the strategies. Figure 6.2 represents the absolute number of types of services found for several strategy configurations. The maximal number of types of service that could be found are 36 (100%). Unfortunately there are no significant differences in the result sets, except from the strategy Yahoo Related Tags. This bad performance is discussed later in the Subsection 6.2.3 Yahoo Related Tags strategy.

The reason for the similarity of all the results could be the small set of test data. Testing the SOCRADES middleware with such a small set of types of services does not lead to clearer deviations.

Nevertheless we dare to propose an optimal configuration. We would suggest using the Wikipedia strategy. Figure 6.2 does not show a benefit of using a combination of all Wikipedia strategies therefore we recommend applying the simple strategy without extensions only.

In addition we suggest defining the Yahoo strategy as the alternative augmentation strategy. **Thus the configuration is the chain of the strategies Wikipedia and Yahoo.** The reasons why this chain is helpful is explained in the next Subsection 6.2.2 Wikipedia strategy.

We prefer a chain of strategies to a combination because of the following reasons:

- The performance of the Query Augmentation Module falls in case of the strategy combination because of the increase of web accesses.
- The current evaluation could not prove that a combination increases the success.

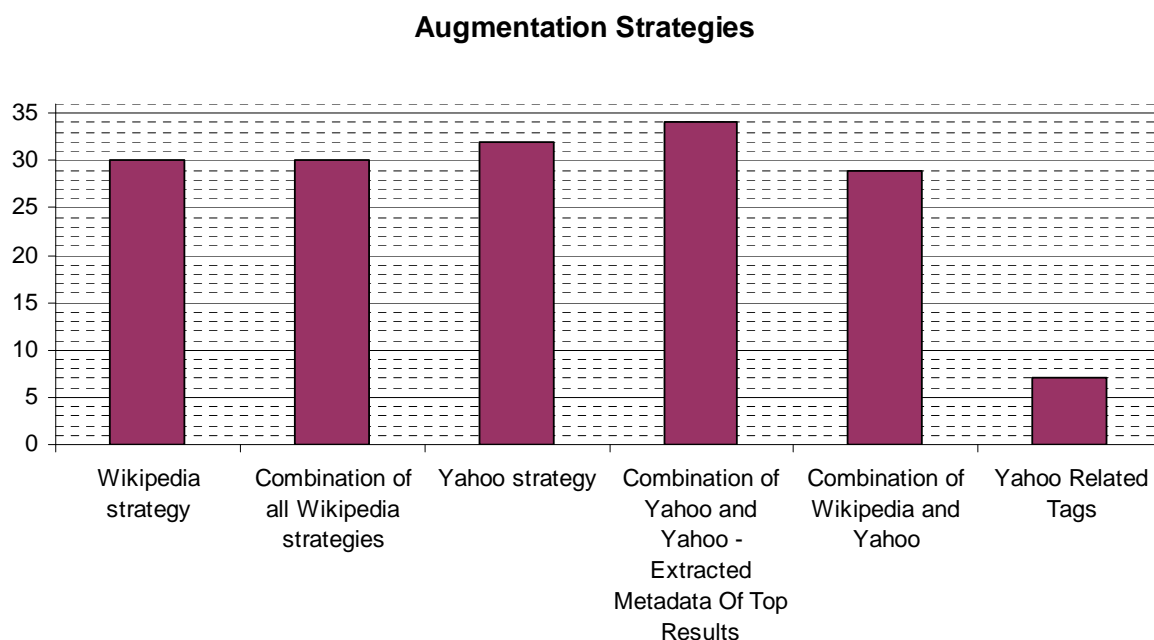


Figure 6.2 Augmentation strategies

6.2.2 Wikipedia strategy

The Wikipedia strategy is the one based on human generated text and as proved in the previous Section it is a useful strategy for the Query Augmentation. As already mentioned in 6.1.1 Data collection we practically evaluated the Query Augmentation Module with the 10 search terms defined at the very beginning. In fact much more search terms will be used to find web services offered by smart devices. So we collected some more search terms for the smart devices of the test data and extended the evaluation of the Wikipedia strategy.

We performed some tests with the additional search terms. For the first tests cases the Query Augmentation Module was configured as proposed in the previous Section. The strategy Wikipedia and as alternative strategy the Yahoo strategy was defined. We expected only a minimal variation of the result compared to the queries with the predefined search terms and so it was. About 90% of the services have been found.

In a second turn of evaluation we performed the tests by configuring single augmentation strategies and did work without combinations or chains. Short after starting the tests for the Wikipedia strategy, we realized that lots of queries do return neither augmented keywords nor types of services because there are no Wikipedia articles for the search terms. Table 6.3 displays some examples of additional search terms. The signs after the term indicate whether a Wikipedia article exists (✓) or not (✗).

| Device | Predefined | Other Search Terms | | |
|----------------------------------|-----------------|---------------------------------|---------------------------|--------------------|
| | Search Terms | | | |
| Brain Surgery Smart cabinet | RFID ✓ | smart shelf ✗ | cupboard ✓ | shelf ✓ |
| Connected product shelf | RFID ✓ | smart shelf ✗ | smart dressing room ✗ | |
| Smart Traffic Counter Service | sensor ✓ | vehicle traffic controller ✗ | traffic counter ✓ | traffic analyzer ✗ |
| Turning Device | conveyor belt ✓ | dispatcher ✓ | supply chain controller ✗ | star orienter ✗ |
| Water meter | smart meter ✓ | boiler ✓ | water measurement ✗ | water meter ✓ |

Table 6.3 Search terms

A complete list of all the collected search terms is attached in appendix B Smart Devices.

Five neutral persons provided as 61 search terms. The outcome is about 4 search terms assigned to each one of the 16 smart devices. We launched the Query Augmentation with 61 queries and found out that only for 33 (54.1%) of the search terms a Wikipedia article exists. The result is illustrated in Figure 6.3.

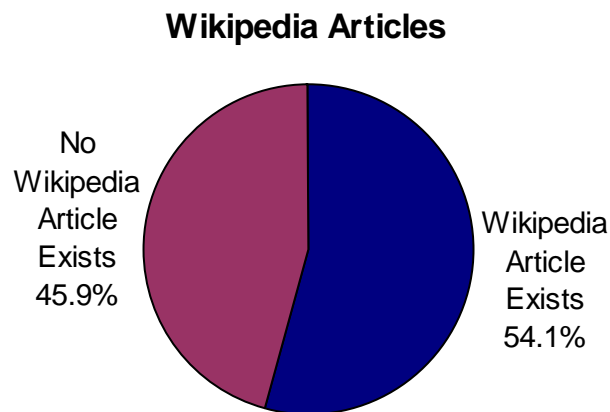


Figure 6.3 Existence of Wikipedia Articles

For about 50% of the search terms used to find services no Wikipedia article does exist. That again means that the Query Augmentation fails in about 50% for all the strategies based on Wikipedia.

This result raises the question if it still makes sense to use the Wikipedia strategies or if these strategies should be removed from the Query Augmentation Module.

In spite of the bad result we still think that the principle of the Wikipedia strategies is good. For the following reason we would still recommend focusing on Wikipedia:

1. Wikipedia (the English version) is growing about 2000 articles per day [53]. With the growth of Wikipedia the chance of finding articles for the search terms is increasing too.

2. We consider an improvement of the implementation. Right now the Wikipedia strategies are checking if an article exists for a search term or not. We could develop a more enhanced processing, which is analyzing the search term more detailed. Assume the search term “supply chain controller”. Instead of using the query as one search term, we could treat it as several search terms (“supply chain” and “controller”) and we could launch the Query Augmentation with two search terms. The Wikipedia strategies would gain flexibility and the result of the Wikipedia strategy probably would improve.

6.2.3 Yahoo Related Tags strategy

Yahoo! describes its My Web 2.0 Web Services: Related Tags as following: “The Related Tags service lets you find tags that appear together on URLs. For example, if a URL is tagged with 'yahoo' and 'music', a search for the tag 'yahoo' will return 'music' as a related tag.” [8] After performing some test online, this Yahoo! Search Web Service seemed to be a perfect strategy for the Query Augmentation Module. A complete web service offered by Yahoo! and even the augmentation logic has been implemented by Yahoo!, so we integrated this web service and called the strategy Yahoo Related Tags.

Unfortunately the evaluation of this strategy did not return the expected success and we can not recommend using the strategy Yahoo Related Tags for query augmentation within the SOCRADES Application Service Catalogue.

We performed the evaluation of the strategy with a query of 10 enriched keywords in this case also called 10 related tags of the search term. We did not add the initial search term to the query and therefore tried to find the types of service with the related tags only. Table 6.4 shows the result, which is the worst result we got over all the evaluation. Only 19% of the types of services have been found.

| Search term | RFID | temperature | sensor | conveyor belt | access control | gripper | vibration | smart meter | webcam | TOTAL | Found | False Pos. |
|-------------|------|-------------|--------|---------------|----------------|---------|-----------|-------------|--------|-------|-------|------------|
| Found | 0 | 3 | 1 | 0 | 0 | 3 | 0 | 0 | 0 | 7 | 19% | |
| False Pos. | 2 | 0 | 2 | 0 | 3 | 0 | 2 | 0 | 0 | 9 | | 3% |

Table 6.4 Yahoo Related Tags - Summary

Consequently we had a closer look at the related tags returned from the Yahoo Web Service and we realized that lots of tags apply to a context that does not make sense for SOCRADES. Table 6.5 represents a sample of search terms and their related tags.

| Search term | temperature | sensor | access control | vibration | webcam |
|-------------|-------------|----------|----------------|------------|------------|
| 1. | weather | laptop | security | energy | dating |
| 2. | conversion | powermac | locksmith | spiritual | live |
| 3. | reference | motion | technology | meditation | web design |

| | | | | | |
|-----|-----------|------------|---------------|---------------|--------------------------|
| 4. | celsius | appel | 24 hour | yoga | business directory |
| 5. | travel | gebrauchte | emergency | neurogenesis | algarve radio |
| 6. | science | macbook | alarm system | karma secrets | algarve free classifieds |
| 7. | tools | macintosh | cards | erik valdman | world |
| 8. | online | macbooks | credit | oneness | seo optimization |
| 9. | bed | notebook | alarm systems | stress | love |
| 10. | converter | 2007 | internet | consciousness | directory algarve |

Table 6.5 Yahoo Related Tags

For the strategy Yahoo Related Tags we arrived at the conclusion that this great idea of the Yahoo! Web Services developers cannot be used for the SOCRADES middleware and it should be removed from the Query Augmentation Module.

We are not pleased with the result of the Yahoo! Web Service and we feel confirmed in our approach of consuming web resources, analyzing them within the Query Augmentation Module and then composing the result set by ourselves. In contrast, the result of the Yahoo! Related Tags has been used directly and without internal processing.

6.3 Discussion

First we are again going back to the research questions and answer them in brief.

Using Query Augmentation in the SOCRADES Application Service Catalogue does improve the success of the search requests. This fact could be verified throughout the whole evaluation (see Figure 6.1 and Table 6.2).

The second research question, if strategies based on human generated texts do lead to better results cannot be answered here. It is very difficult to make a specific statement as the results of all the strategies are very close to each other and the set of test data is rather small. We would therefore recommend reevaluating the different strategies with a much larger set of test data. At the same time we should think about ameliorations within the evaluation process:

- The implementation of the strategies also influences the result. At the end of Subsection 6.2.2 Wikipedia strategy we recommended to improve the Wikipedia strategies and this will change the result too.
- Until now only the number of search terms and the augmentation strategies has been considered as variable for the evaluation. Though there exist other parameters to configure for the individual strategies. E.g. the weights of the extended Wikipedia strategies or the number of web search results of Yahoo. Before we are able to make a reliable statement we have to optimize all the variables, i.e. configuration parameters, of the Query Augmentation Module.

The same arguments have to be kept in mind when answering the last research question by defining the optimal strategy or strategies. At the present state of the evaluation we assume a chain of the strategies Wikipedia and Yahoo as the best configuration for the Query Augmentation Module. This may change if all configuration parameters within the module would be optimized.

In general we arrive at the conclusion that we could prove the usefulness of Query Augmentation and we found out that a lot of test cases are missing in the current evaluation.

So we conclude that a much larger scale of experiments as well as optimization of parameters has to be done and in certain cases first improve the implementation of the individual strategies. Future work includes testing with more realistic test data. We observed that additional Wikipedia concepts did not help a lot the basic algorithm, more understanding is required on how and when these additional features should be used. The next Section gives some inputs that have to be considered for future evaluation of the Query Augmentation Module too.

6.3.1 Future evaluation

The first and most important thing that has to be considered is the configuration of the augmentation strategies. More combinations of strategies have to be tested, e.g. compare the different Wikipedia strategies among each other. Until now only the simple Wikipedia strategy and the combination of all Wikipedia strategies has been evaluated. We could imagine evaluating the following combinations and finally we would find the best configuration within the Wikipedia strategies e.g. the combination of the strategies Wikipedia and Wikipedia Links, or the strategies Wikipedia and Wikipedia Backlinks. In the worst case we may be able to prove that the extended Wikipedia strategies do not lead to any benefit.

And we should consider optimizing the implementation of the current strategies according to the current results and consequences before evaluating the Query Augmentation again.

As already mentioned until now we only thought about two configuration parameters but we should optimize all the parameters available. The improvements concerning the strategies are described in the Section above. But the number of keywords has not been tested optimally too. The following numbers have been concerned in the evaluation: 1, 5, 10, 20, and 50. We saw that the range from 1 to 10 has been the most interesting one in terms of the optimal number. Without testing this range from 1 to 10 more detailed we decided that 5 keywords are optimal for the enrichment. But maybe it is worth to perform more testing on 2, 3, 4, etc. keywords.

Performance

In order to prove the assumption that chaining strategies is better than combining strategies we should add performance measurements to the evaluation. We assume that the chaining takes less time as there are less web accesses needed. The easiest way to prove that would be a component that is registering the time each strategy needs in order to find its enriched keyword list. Such a component could easily be added to the Query Augmentation UI.

7

Conclusion

| | |
|--|-----------|
| 7.1 Outlook | 62 |
| 7.2 Final Thoughts about Query Augmentation | 62 |
| 7.3 Realisation of the master thesis | 63 |

7.1 Outlook

A lot of smart devices that are offering information and services exist nowadays in the industry. The challenge is to find them and to make use of them. The challenge of managing smart devices is not yet optimally solved. A possible solution to that is SOCRADES and its Application Service Catalogue designed to find smart devices easily. Already the simple implementation of the Query Augmentation included in the Application Service Catalogue seems to play an important role for the search of services. It has been shown that the Query Augmentation Module is important for SOCRADES and we are sure that it will be used, improved and expanded in the future.

The current version of the Query Augmentation Module within the SOCRADES middleware meets all the requirements. The current version offers a stable, basic solution with a very flexible architecture. The open architecture of the Query Augmentation should encourage the project team to add some more sophisticated query enrichment strategies.

7.2 Final Thoughts about Query Augmentation

The main goal of this master thesis was the elaboration of the Query Augmentation Module that is part of the SOCRADES middleware developed by SAP. Several augmentation strategies have been developed and used for answering our research questions. We were able to demonstrate that Query Augmentation does increase the chance of success of the search of services. Contrary to our assumptions we were not able to prove that strategies based on human generated texts, such as our Wikipedia strategies, lead to better results than others, e.g. strategies based on common search engines such as Yahoo! Web Search. Amongst our developed strategies we propose a chain of strategies, that is Wikipedia strategy and the Yahoo! Web Search strategy as alternative one, as the best strategy configuration. These results are based on our own evaluation which is based on a small data sample. For a

refinement of the result more experiments based on more realistic test data have to be performed.

During the elaboration of the Query Augmentation Module we realized that the implementation and elaboration of this module not only contains the development of the concrete augmentation strategies, but also the architecture of the component, the design the UI, and the integration into the SOCRADES project. A lot of time has been spent with the analysis of advanced enrichment strategies, such as approaches that are near to the semantic web. But finally most of the time has been used for the elaboration of the flexible design, the UI and the integration into the SAP project environment. The implemented strategies are not very elevated and we had to abstain from advanced query enrichment strategies using the semantic web. Nevertheless, we achieved our goals and proved that Query Augmentation does help the user finding web services on smart devices. In addition we collected a lot of ideas for the improvement of the current Query Augmentation Module.

7.3 Realisation of the master thesis

I would like to use this section to describe my personal experiences in connection with the master thesis. Therefore this section is written from the author's point of view. The elaboration of the thesis in conjunction with an internship at the SAP Research Center in Zurich was a great opportunity for me to get an insight into research on the one hand and to participate in a real software development project on the other hand. The SAP Research Center gave me an overview about industrial research activities and at the same time I got to know the duties and responsibilities of Ph.D. students. This helped me for my own decision making process whether I would follow the academic way as a Ph.D. student or whether I would change into the industry and start working as an information scientist after the final degree. Even though I got a good impression about the research I finally decided to start working for an internet agency and trying to apply the knowledge established at the university and at the SAP Research Center.

My internship, which was my master thesis at the same time, was divided into four parts. In the warm-up phase I got to know the SOCRADES project, the SAP Research Center and the SAP development and product environment. SOCRADES was my first big software development project with active participation. In a second phase I was gathering technical skills by going through SAP internal and external Java EE tutorials. I was already familiar with the Java EE technologies from courses at university. But in contrast to the knowledge about engineering concepts I needed at university, the practices have been the important part of Java EE for the master thesis. In addition to gathering Java EE experience a first implementation of the Query Augmentation Module belonged to the second phase too. I got the opportunity to do that implementation within a one week boot camp of the whole SOCRADES team. That was a good chance to benefit from the expert knowledge of the experienced team members. Once we had a running Query Augmentation we added a UI, evaluated Query Augmentation, and improved and corrected the strategies. That was the third, longest, and most intensive part of my master thesis. The final phase was then the creation of this report. The project schedule presented a good fundament for the master thesis and two third of the projected have passed as planed. Only the evaluation especially the generation of the test data and the writing of the report turned out to be more time consuming than planned.

For many reasons this master thesis was an instructive experience for me. The help of the team members, the SAP researchers in general and the supervision from both parties, the

University of Fribourg and the SAP Research Center, was great. I appreciate that the Department of Computer Science at the University of Fribourg supports master thesis in collaboration with other research institutions and the industry.

.



Project schedule

Phase 0: Warmup: 05.05.08 – 16.05.08

- Reading the Specification and some related work.
- Installing and getting in touch with SAP Web Applications Server (SAP Web AS) and Netweaver Developer Studio.

Phase 1: Technical Skills gathering and First Implementation: 19.05.08 -30.07.08:

- Minimal implementation of the Query Augmentation module with at least one Query Strategy.

Phase 2: Improvements, User Interface and Product Integration: 30.07.08 – 01.11.08

- Improving the module.
- Creating a “User Interface” (topmost module on Figure 2) by integrating the implementation to an SAP product (either SAP MII, SAP Business By Design (ERP) or Netweaver Dev Studio).

Phase 3: Documentation, Report and Literature Review: 01.11.08-30.11.08

Writing a scientific report (a Master Thesis, ~60 pages) including a review of the related work.

B

Smart Devices

| | Device | Details | Search Term | Other Search Terms | | |
|---|-------------------------------|---|------------------------------------|----------------------------|-------------------------|------------------|
| 1 | MR RFID Reader | http://www.tagsysrfid.com/systems/upload/MedioP101_Eth_Datasheet.pdf also check: http://www.tagsysrfid.com/solutions/textile-services.cfm http://www.tagsysrfid.com/systems/hf-readers.cfm (2nd reader) | rfid | rfid reader | | |
| 2 | Smart cabinet | http://www.tagsysrfid.com/systems/upload/TAGSYS-SC400-Datasheet-24Sep07.pdf also check: http://www.tagsysrfid.com/systems/hf-application-stations.cfm (next to lowermost) | rfid | smart shelf | cupboard | shelf |
| 3 | Connected product shelf | http://www.checkpointeurope.com/app/?topsectionid=11004&topsection=productgroup&sectionid=&section=&subsectionid=&subsection=&bottomsectionid=&bottomsection=&locale=eu&page=groupproduct&id=1762 | rfid | smart shelf | smart dressing room | |
| 4 | MyWeather | http://www.wetterdirekt.com/?key=produkte (1st product) | temperature | weather station | home weather station | |
| 5 | SunSPOT | http://www.sunspotworld.com/products/ (scroll down) | temperature sensor vibration | sunspot | smart sensor | |
| 6 | Smart Traffic Counter Service | http://www.sensourceinc.com/TC-MS30.htm | sensor | vehicle traffic controller | traffic counter | traffic analyser |
| 7 | Turning Device | http://www.flexlink.com/resources/10000/pdf/10-14083.pdf | conveyor belt | dispatcher | supply chain controller | star orienter |

| | Device | Details | Search Term | Other Search Terms | | |
|----|----------------------------|--|-----------------|---------------------------|-------------------|-------------------------|
| 8 | Time and Attendance System | http://www.kaba.ch/en/Products-Solutions/Time-Attendance/9882_9882/time-attendance.html | access control | access controller gate | security door | roatating door |
| 9 | Personal Mechanic Arm | http://www.lynxmotion.com/Product.aspx?productID=160&CategoryID= | gripper | grip | claw | |
| 10 | Smart Robotic Arm | http://www.lynxmotion.com/Product.aspx?productID=2&CategoryID=2 | gripper | traffic controller | flow controller | supply chain controller |
| 11 | MICA | http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf | vibration | sensor board | | |
| 12 | Water meter | http://www.smartmeter.co.uk/product.php?prod_id=1 | smart meter | boiler | water measurement | |
| 13 | Energy Ticker | http://www.mercateo.com/p/102-125320(2d)BP/ENERGIEKOSTENMESSG_ENERGY_MONITOR_3000.html | smart meter | electricity sensor | | |
| 14 | WebCam | http://www.logitech.com/index.cfm/webcam_communications/webcams/devices/3480&cl=us,en | webcam | webcam | face recogition | |
| 15 | Ubisense location sensor | http://www.ubisense.net/media/pdf/Ubisense%207000%20Series%20Sensor.pdf http://www.ubisense.net/content/66.html | location sensor | sensor | | |
| 16 | People Counting Device | http://www.sensormatic.com/Products/SmartEAS/OverheadPeopleCounting.aspx http://www.sensormatic.com/SensormaticGetDoc.aspx?FileID=16989 | location sensor | people counter | crowd counter | |

C

Questionnaire for User Study

SOCRADES – Device Simulation

User study of: Bettina Dober (bettina.dober@sap.com) supervised by Dominique Guinard (dominique.guinard@sap.com)

Background Information: SOCRADES

On the one hand, enterprises manufacturing any kind of goods require agile production technology to be able to fully accommodate their customers' demand for flexibility. On the other hand, Smart Objects, such as networked intelligent machines or tagged raw materials, exhibit ever increasing capabilities, up to the point where they offer their smart behavior as web services. The two trends towards higher flexibility and more capable objects will lead to a service-oriented infrastructure where complex processes will span over all types of systems — from the backend enterprise system down to the Smart Objects. To fully support this, the SOCRADES integration architecture can serve the requirements of future manufacturing and industry. It provides generic components upon which sophisticated production and monitoring processes can be modeled.

The EU SOCRADES project gathers 16 different European industrial and academic partners (Siemens, Schneider Electric, ABB, ARM, Flexlink, etc.), and its primary objective is to develop a design, execution and management platform for next-generation industrial automation systems, exploiting the Service Oriented Architecture paradigm both at the device and at the application level. SAP's role in this project is to build a software infrastructure to integrate smart devices into high-level business processes and applications, such as SAP Enterprise Resource Planning systems.

Project's website: www.socrades.eu

Project's demonstration video: <http://www.youtube.com/watch?v=K8OtFD6RLMM>

SOCRADES Simulation

For setting up the SOCRADES simulation and demonstration environments, virtual smart devices have to be described and created. As a participant to this user study you will provide highly valuable help towards the generation of these smart devices.

Introduction

The implementation of the software infrastructure of SOCRADES is almost finished and has to be evaluated now. As there is no network of real smart devices available yet, we want to test the implementation with virtual smart devices first. A simulation framework for smart devices already exists. And we now need to create virtual smart devices that are offering web services.

We want to set up a database of smart devices described by people with different background knowledge and people that do not know much about the implementation. With this approach the quality of the data is ensured.

Sample Use Case

First a presumable use case of the SOCRADES integration architecture is described here.

One wants to wash clothes now. All washing machines are smart devices that are offering several web services such as services to get the state or a list of washing programs of the washing machine. Thus, one can use a SOCRADES application to search for web services of nearby washing machines.

The user, looking for the washing machine, only has to search for “washing machine” and he may specify his context (e.g. current position of himself). The user is setting up his request in the SAP Enterprise Resource Planning system running on top of the SOCRADES Smart Item Infrastructure. The SOCRADES middleware is executing the request and returns the identified service instances of the nearest washing machines. The services can then be used to get the state of a washing machine or to find out if a desired washing program is offered.

Your task

You have to write the description of the **Smart Shelf & Smart Rail**

(<http://www.checkpointeurope.com/app/?topsectionid=11004&topsection=productgroup§ionid=§ion=&subsectionid=&subsection=&bottomsectionid=&bottomsection=&locale=eu&page=groupproduct&id=1762>).

Please read the online resources carefully, as it will serve you as a basis for describing the services.

Assume that you are the manufacturer of this smart device and you know need to described the services offered by this smart device. This description is then going to be shipped with the device and used for discovering as well as searching for the services it offers.

The indicated web source will help you with general information about your smart device. It is probable that the device on this page does not really offer web services, but it is a device which could potentially offer such services. Thus, your task it to think about the services it could offer and to describe them.

Attention: Your smart device does not refer to the example of the washing machine. Please imagine your own use case.

Thank you for completing **the following form:**

Device description

| Manufacturer / Model Information | Description |
|---|---|
| Manufacturer | Checkpoint Systems |
| Manufacturer URL | http://www.checkpointeurope.com |
| Name of the model | Smart Shelf & Smart Rail |
| URL to a description of the model | http://www.checkpointeurope.com/app/?topsectionid=11004&topsection=productgroup&sectionid=&section=&subsectionid=&subsection=&bottomsectionid=&bottomsection=&locale=eu&page=groupproduct&id=1762 |
| Friendly name of the device (e.g. Washing machine and tumbler) | Connected product shelf. |
| Brief description of the device | Shelf which provides remote access to electronic information about inventory and other relevant data about contained products or scheduled refills. |

| Services Information of Service 1 | Description |
|---|--------------------|
| Name of service 1 (e.g. Washing Machine service) | |
| Friendly description of service 1 (e.g. A fully-featured washing machine service that you can use to wash all your clothes.) | |
| Name of operation 1 (e.g. cycle(time, speed)) | |
| Friendly description of operation 1 (e.g. Represents a speed and wash time) | |
| Name of operation 2 (e.g. cycle(time, speed)) | |
| Friendly description of operation 2 (e.g. Represents a speed and wash time) | |
| Friendly description of operation 3 (e.g. Represents a speed and wash time) | |

C Questionnaire for User Study

Name of operation 3
(e.g. cycle(time, speed))

Friendly description of operation 3
(e.g. Represents a speed and wash time)

Name of operation 4
(e.g. cycle(time, speed))

Friendly description of operation 4
(e.g. Represents a speed and wash time)

Name of operation 5
(e.g. cycle(time, speed))

Friendly description of operation 5
(e.g. Represents a speed and wash time)

Services Information of Service 2 Description

Name of service 2
(e.g. Washing Machine service)

Friendly description of service 2
(e.g. A fully-featured washing machine service that you can use to wash all your clothes.)

Name of operation 1
(e.g. cycle(time, speed))

Friendly description of operation 1
(e.g. Represents a speed and wash time)

Name of operation 2
(e.g. cycle(time, speed))

Friendly description of operation 2
(e.g. Represents a speed and wash time)

Name of operation 3
(e.g. cycle(time, speed))

Friendly description of operation 3
(e.g. Represents a speed and wash time)

C Questionnaire for User Study

Name of operation 4
(e.g. cycle(time, speed))

Friendly description of operation 4
(e.g. Represents a speed and wash time)

Name of operation 5
(e.g. cycle(time, speed))

Friendly description of operation 5
(e.g. Represents a speed and wash time)

D

WSDL File

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <wsdl:definitions name="SmartShelfAndSmartRail"
3   targetNamespace="http://www.checkpointeurope.com/ConnectedPro
ductShelf"
4   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
5   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
6   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap12/"
7   xmlns:wse="http://schemas.xmlsoap.org/ws/2004/08/eventing"
8
  xmlns:tns="http://www.checkpointeurope.com/ConnectedProductShelf
">
9   <wsdl:documentation>Shelf which provides remote access to
electronic information about inventory and other relevant data
about contained products or scheduled
refills.</wsdl:documentation>
10
11     <!--
12     TYPES
13     -->
14
15     <wsdl:types>
16       <xsd:schema
targetNamespace="http://www.checkpointeurope.com/ConnectedProduc
tShelf"
xmlns:tns="http://www.checkpointeurope.com/ConnectedProductShelf
"/>
17     </wsdl:types>
18
19     <!--
20     MESSAGES
21     -->
22
23     <wsdl:message name="HasProductRequestMsg" >
24       <wsdl:part name="descriptionIn" type="xsd:string" />
25     </wsdl:message>
26     <wsdl:message name="HasProductResponseMsg" >
27       <wsdl:part name="availabilityOut" type="xsd:boolean"
/>
28     </wsdl:message>
29     <wsdl:message name="HasBrandRequestMsg" >
30       <wsdl:part name="nameIn" type="xsd:string" />
31     </wsdl:message>
32     <wsdl:message name="HasBrandResponseMsg" >
33       <wsdl:part name="availabilityOut" type="xsd:boolean"
/>
34     </wsdl:message>
35     <wsdl:message name="GetProductCountRequestMsg" >
36       <wsdl:part name="descriptionIn" type="xsd:string" />
37     </wsdl:message>
38     <wsdl:message name="GetProductCountResponseMsg" >
39       <wsdl:part name="countOut" type="xsd:int" />
40     </wsdl:message>
41     <wsdl:message name="GetShopInformationMsg" >
42       <wsdl:part name="infoOut" type="xsd:string" />
43     </wsdl:message>
44     <wsdl:message name="GetNextAvailableDateRequestMsg" >
45       <wsdl:part name="productDescriptionIn"
type="xsd:string" />
46     </wsdl:message>
47     <wsdl:message name="GetNextAvailableDateResponseMsg" >
```

```
48         <wsdl:part name="refillDateOut" type="xsd:date" />
49     </wsdl:message>
50
51     <!--
52         PORT TYPES
53     -->
54     <wsdl:portType name="SmartShelfCustomerService"
55 wse:EventSource="true">
56         <wsdl:documentation>
57             Service provided to a customer by a connected Smart
58             Shelf which
59                 lets you identify the location and availability
60                 of products by
61                 various criteria remotely.
62                 </wsdl:documentation>
63                 <wsdl:operation name="HasProduct">
64                     <wsdl:documentation>
65                         Indicates whether a product whose name or
66                         description
67                         contains the given string is available
68                         on this shelf and
69                         includes product information which
70                         includes price, etc.
71                     </wsdl:documentation>
72                     <wsdl:input message="tns:HasProductRequestMsg" />
73                     <wsdl:output
74 message="tns:HasProductResponseMsg" />
75                     </wsdl:operation>
76                     <wsdl:operation name="HasBrand">
77                         <wsdl:documentation>
78                             Indicates whether at least one product
79                             of a given brand
80                             name is available on this shelf and
81                             includes product
82                             information which includes price, etc.
83                         </wsdl:documentation>
84                         <wsdl:input message="tns:HasBrandRequestMsg" />
85                         <wsdl:output
86 message="tns:HasBrandResponseMsg" />
87                         </wsdl:operation>
88                         <wsdl:operation name="GetProductCount">
89                             <wsdl:documentation>
90                                 Returns the current count of products that match
91                                 the
92                                 given description.
93                             </wsdl:documentation>
94                             <wsdl:input
95 message="tns:GetProductCountRequestMsg" />
96                             <wsdl:output
97 message="tns:GetProductCountResponseMsg" />
98                             </wsdl:operation>
99                             <wsdl:operation name="GetShopInformation">
100                                 <wsdl:documentation>
101                                     Returns information about the shop which
102                                     owns this shelf.
103                                 </wsdl:documentation>
104                                 <wsdl:documentation>
105                                     That is, information such as the address
106                                     or general
107                                     information about available products.
108                                 </wsdl:documentation>
```

```
92         <wsdl:output message="tns:GetShopInformationMsg"
/>
93     </wsdl:operation>
94         <wsdl:operation name="GetNextAvailableDate">
95             <wsdl:documentation>
96                 Returns a date to indicate when the refill of a
97                 given
98                     (out-of-stock) product is to be
99                 expected.
100             </wsdl:documentation>
101             <wsdl:input
message="tns:GetNextAvailableDateRequestMsg" />
102             <wsdl:output
message="tns:GetNextAvailableDateResponseMsg" />
103         </wsdl:operation>
104     </wsdl:portType>
105     <!--
106     BINDINGS
107     -->
108     <wsdl:binding name="SmartShelfCustomerService"
type="tns:SmartShelfCustomerService">
109         <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"
/>
110         <wsdl:operation name="HasProduct">
111             <soap:operation soapAction="" style="document"/>
112             <wsdl:input>
113                 <soap:body use="literal" />
114             </wsdl:input>
115             <wsdl:output>
116                 <soap:body use="literal" />
117             </wsdl:output>
118         </wsdl:operation>
119         <wsdl:operation name="HasBrand">
120             <soap:operation soapAction="" style="document"/>
121             <wsdl:input>
122                 <soap:body use="literal" />
123             </wsdl:input>
124             <wsdl:output>
125                 <soap:body use="literal" />
126             </wsdl:output>
127         </wsdl:operation>
128         <wsdl:operation name="GetProductCount">
129             <soap:operation soapAction="" style="document"/>
130             <wsdl:input>
131                 <soap:body use="literal" />
132             </wsdl:input>
133             <wsdl:output>
134                 <soap:body use="literal" />
135             </wsdl:output>
136         </wsdl:operation>
137         <wsdl:operation name="GetShopInformation">
138             <soap:operation soapAction="" style="document"/>
139             <wsdl:output>
140                 <soap:body use="literal" />
141             </wsdl:output>
142         </wsdl:operation>
143         <wsdl:operation name="GetNextAvailableDate">
```

```
144         <soap:operation soapAction="" style="document" />
145         <wsdl:input>
146             <soap:body use="literal" />
147         </wsdl:input>
148         <wsdl:output>
149             <soap:body use="literal" />
150         </wsdl:output>
151     </wsdl:operation>
152 </wsdl:binding>
153
154
155 <!--
156     Services (Not used by the toolkit)
157 -->
158 <wsdl:service
159     name="ConnectedProductShelf_SmartShelfCustomerService">
160         <wsdl:port name="SmartShelfCustomerService"
161         binding="tns:SmartShelfCustomerService" >
162             <soap:address
163             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
164             location="http://www.checkpointeurope.com/DPWS/samples/Connected
165             ProductShelf/SmartShelfCustomerService" />
166         </wsdl:port>
167     </wsdl:service>
168 </wsdl:definitions>
```

E

Deployment with NWDS

| | |
|--|-----------|
| Preconditions | 79 |
| NWDS workspace | 79 |
| SAP NetWeaver Application Server | 80 |
| NWDS preferences..... | 80 |
| Deployment process | 81 |
| Add and Remove Projects | 81 |
| Publish projects | 82 |

The Query Augmentation Module and the Query Augmentation UI are part of the SOCRADES Application Service Catalogue component, which is implemented in a Java EE application with the SAP NetWeaver Developer Studio (NWDS). In order to deploy the application the according Java enterprise archive has to be transferred to a SAP NetWeaver Application Server (SAP AS).

A SAP NetWeaver Composition Environment 7.1 SR5 – Trial Version that includes the SAP NetWeaver Application Server (Java EE Edition) and the SAP NetWeaver Developer Studio (with Java EE support and based on Eclipse Europa) is available on [40]. Please note that the current version of the NetWeaver Environment is supporting the J2SE Runtime Environment 5.0 and not yet 6.0.

This Section contains all necessary information to deploy and undeploy the Application Service Catalogue. We assume that you are familiar with the SAP software development environment and that you have the required access rights.

Preconditions

All the following conditions must be fulfilled for the deployment of the Application Service Catalogue, the `SII_application_service_catalog_EAR` project.

NWDS workspace

Following projects have to be loaded from SOCRADES' SVN directory to the NWDS workspace:

- SII_application_service_catalog_EAR
- SII_application_service_catalog_ejb
- SII_application_service_catalog_ui
- SII_commonDataDic
- SII_CommonDataStructures

All of these projects are stored in the SVN directory IntegrationArchitecture.

The workspace may contains additional projects. That does not cause problems. The above list just shows the minimal requirement of projects.

SOCRADES subversion directory:

Source code projects can be downloaded from the SOCRADES SVN directory with the according dialog.

1. Choose File > New > Other... > SVN > Checkout projects from SVN then click Next.
2. Choose Use existing repository location and enter <https://forge.sap.corp/svn/socrades> then click Next
3. Follow the dialog and remember that SOCRADES projects are stored in the folder IntegrationArchitecture.


[54] offers more information about SOCRADES and Subversion.

SAP NetWeaver Application Server

The application server and the database server have to run properly. In addition the following SOCRADES projects have to be deployed on the application server.

- SII_service_impl_repository_EAR
- SII_service_monitor_EAR
- SII_discovery_EAR

Within the Deploy perspective of the NWDS you can control, which projects are deployed.

To open a perspective choose Window > Open Perspective > Other... and select  Deploy then click OK.

NWDS preferences

The application server, on which the Application Service Catalogue wants to be deployed, has to be configured in the NWDS.


Choose Window > Preferences > SAP AS Java and register the SAP instance.

Deployment process

If all the conditions of the previous Section are met, the Application Service Catalogue project (SII_application_service_catalog_EAR) can be deployed.

Add and Remove Projects

This ‘Add and Remove Projects’ dialog is used for the initial deployment of a project and for the undeployment.

For the deployment you will need the view Servers. If you are working in the Java EE perspective ( Java EE) of the NWDS this view among others is available on the bottom of the NWDS.

1. Go to the view Servers and right click on the server instance then select Add and Remove Projects..., which opens the dialog illustrated in Fig. 1.
2. Add the project SII_application_service_catalog_EAR to Configured projects then click Finish.
3. The deployment starts directly after clicking Finish. It will take a while and as soon as the process is finished a confirmation message pops up.

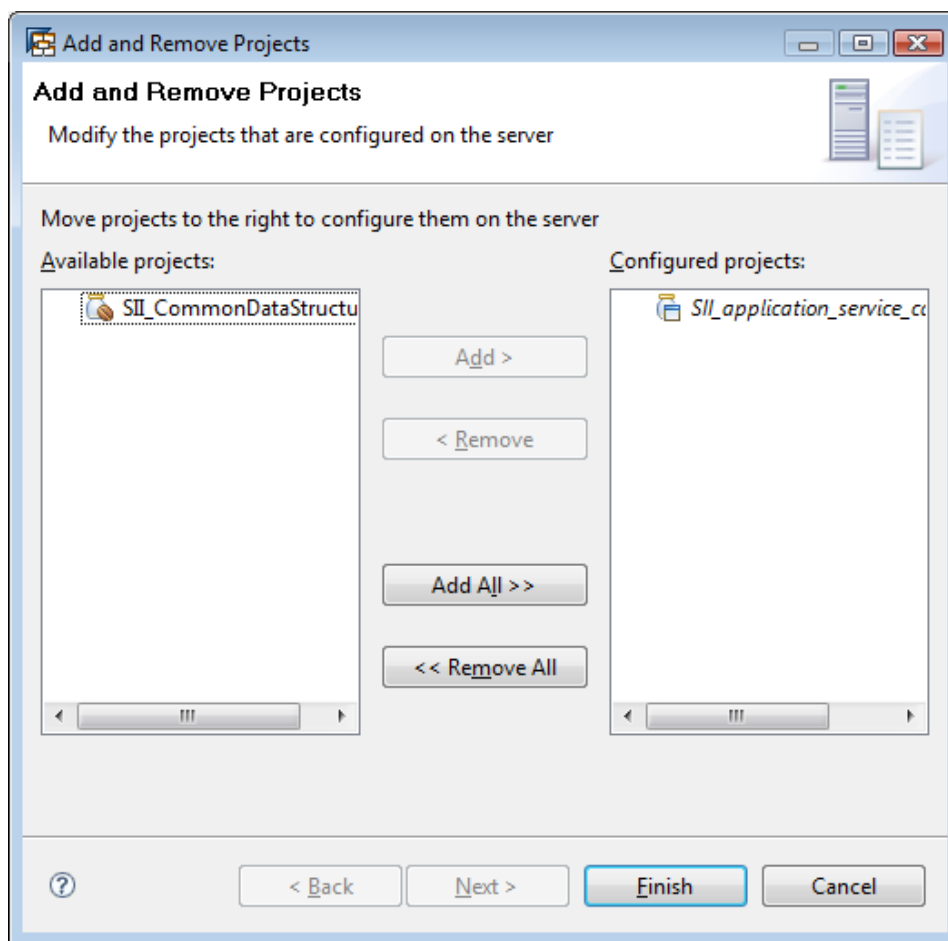


Fig. 1 Add and Remove Projects (NWDS 7.1)

The undeployment works similar to the deployment by removing the project from Configured projects. An alternative variant of the undeployment is offered in the Deploy perspective.

Publish projects

A project that already has been deployed from your workspace is listed in the view Servers and can be redeployed by right clicking the server and selecting Publish.

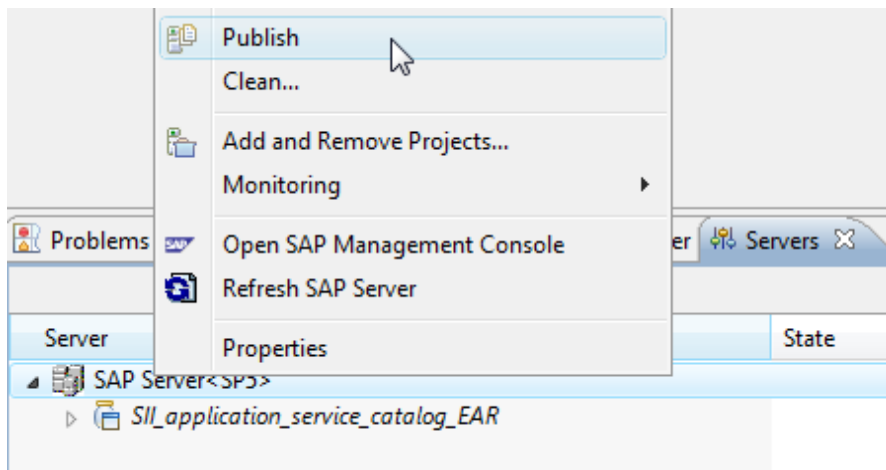


Fig. 2 Publish project (NWDS 7.1)

F

Common Acronyms

| | |
|---------|---|
| ABAP | Advanced Business Application Programming originally Allgemeiner Berichts-Aufbereitungs-Prozessor = general report creation processor |
| API | Application Programming Interface |
| AS | Application Server |
| CEC | Campus-based Engineering Center |
| CRM | Customer Relationship Management |
| CSS | Cascading Style Sheet |
| DPWS | Device Profile for Web Services |
| EJB | Enterprise JavaBeans |
| ERP | Enterprise Resource Planning |
| GUI | Graphical User Interface |
| IDE | Integrated development environment |
| IEEE | Institute of Electrical and Electronics Engineers |
| Java EE | Java Enterprise Edition |
| Java SE | Java Standard Edition |
| JPA | Java Persistence API |
| JSF | JavaServer Faces |
| JSON | JavaScript Object Notation |
| JSP | JavaServer Pages |
| MES | Manufacturing Execution Software |
| MVC | Model, View and Controller |
| NWDS | SAP NetWeaver Developer Studio |

| | |
|----------|--|
| OLE | Object Linking an Embedding |
| OPC-UA | OLE for Process Control – United Architecture |
| PLM | Product Lifecycle Management |
| POJO | Plain Old Java Object |
| REST | Representational State Transfer |
| RFID | Radio Frequency Identification |
| SAP (DE) | Systemanalyse und Programmentwicklung Systeme, Anwendungen und Produkte in der Datenverarbeitung |
| SAP (EN) | System Analysis and Program Development Systems, Applications and Products in Data Processing |
| SCM | Supply Chain Management |
| SKOS | Simple Knowledge Organization System |
| SOA | Service Oriented Architecture |
| SOCRADES | Service Oriented Cross-layer infrastructure for Distributed smart Embedded devices |
| SRM | Supplier Relationship Management |
| SVN | Subversion (software) |
| UI | User Interface |
| URI | Uniform Resource Identifier |
| W3C | World Wide Web Consortium |
| WS(A)N | Wireless Sensor (and Actor) Networks |
| WSDL | Web Service Description Language |

G

D-ROM

CD-ROM content:

- Master thesis report
- User study
 - Summary of smart devices
 - Questionnaires (empty)
 - Questionnaires (returned)
 - WSDL files
 - Results
- Presentation
 - Project presentation
 - SOCRADES movie (short version)
 - 2 demonstration use cases (PDF and wink [55] files)
- Javadoc of the Query Augmentation Module
- Selection of referenced papers

References

- [1] “Socrates.eu” Available at: <http://www.socrates.eu> [Accessed November 12, 2008].
- [2] “Software Engineering Group - Documentation Guidelines” Available at: <http://diuf.unifr.ch/softeng/guidelines.html> [Accessed December 11, 2008].
- [3] “Devices Profile for Web Services” Available at: <http://schemas.xmlsoap.org/ws/2006/02/devprof/> [Accessed November 25, 2008].
- [4] “YouTube - SOCRADES Demo” Available at: <http://www.youtube.com/watch?v=K8OtFD6RLMM> [Accessed November 25, 2008].
- [5] E. Gamma, R. Helm, R.E. Johnson, and J. Vlissides, *Design Patterns. Elements of Reusable Object-Oriented Software.*, Addison-Wesley, 1995.
- [6] “MediaWiki API” Available at: <http://en.wikipedia.org/w/api.php> [Accessed November 25, 2008].
- [7] “Web Search Documentation for Yahoo! Search Web Services ” Available at: <http://developer.yahoo.com/search/web/V1/webSearch.html> [Accessed November 25, 2008].
- [8] “Yahoo Web 2.0 Web Services: Related Tags” Available at: <http://developer.yahoo.com/search/myweb/V1/relatedTags.html> [Accessed November 25, 2008].
- [9] “DBpedia” Available at: <http://dbpedia.org> [Accessed November 25, 2008].
- [10] W. Balke and M. Wagner, “Through different eyes: assessing multiple conceptual views for querying web services,” *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, ACM New York, USA, 2004, pp. 196-205.
- [11] O. Baecker, T. Frenken, F. Geller, D. Guinard, S. Karnauskos, M. Köhler, L. Moreira Sa de Souza, D.K. Nguyen, D. Savio, P. Spiess, and M.V. Trifa, “Specification of the SOCRADES Integration Architecture.”
- [12] “Developer's Guide - Google AJAX Search API - Google Code” Available at: <http://code.google.com/intl/und/apis/ajaxsearch/documentation/> [Accessed December 16, 2008].
- [13] “Query expansion - Wikipedia, the free encyclopedia” Available at: http://en.wikipedia.org/wiki/Query_expansion [Accessed December 12, 2008].
- [14] Z. Zhengyu , X. Jingqiu , R. Xiang , T. Yunyan , and L. Lipei , “Query Expansion Based on a Personalized Web Search Model,” *Semantics, Knowledge and Grid, Third International Conference on*, Shan Xi: IEEE, 2007, pp. 128-133.

- [15] D. Bachlechner and K. Siorpaes, "Harvesting Wiki Consensus -Using Wikipedia Entries as Ontology Elements," *Proceedings of the Workshop on Semantic Wikis at the ESWC2006* , 2006, pp. 54-65.
- [16] Zareen Syed, T. Finin, and A. Joshi, "Wikipedia as an Ontology for Describing Documents," *Proceedings of the Second International Conference on Weblogs and Social Media* , AAAI Press, 2008.
- [17] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine" Available at: <http://infolab.stanford.edu/pub/papers/google.pdf> [Accessed January 20, 2009].
- [18] S. Banerjee, K. Ramanathan, and A. Gupta, "Clustering Short Texts using Wikipedia," Amsterdam: HP Labs, 2007.
- [19] "Resource Description Framework (RDF) / W3C Semantic Web Activity" Available at: <http://www.w3.org/RDF/> [Accessed December 11, 2008].
- [20] "OWL Web Ontology Language Reference" Available at: <http://www.w3.org/TR/owl-ref/> [Accessed December 11, 2008].
- [21] "Microformats" Available at: <http://microformats.org/> [Accessed February 4, 2009].
- [22] "SPARQL Query Language for RDF" Available at: <http://www.w3.org/TR/rdf-sparql-query/> [Accessed December 10, 2008].
- [23] "Yago - A Core of Semantic Knowledge" Available at: <http://www.mpi-inf.mpg.de/~suchanek/downloads/yago/> [Accessed December 11, 2008].
- [24] "DBpedia - Virtuoso SPARQL Query Form" Available at: <http://dbpedia.org/sparql> [Accessed December 13, 2008].
- [25] "OpenLink Universal Integration Middleware - Virtuoso Product Family" Available at: <http://virtuoso.openlinksw.com/> [Accessed December 16, 2008].
- [26] "SKOS Simple Knowledge Organization System - home page" Available at: <http://www.w3.org/2004/02/skos/> [Accessed December 16, 2008].
- [27] B. Dober, "Exploring Query Augmentaton for the SOCRADES Application Service Catalogue, Master thesis, Official webpage, Unjiversit of Fribourg, 2009" Available at: <http://diuf.unifr.ch/softeng/student-projects/completed/dober> [Accessed November 12, 2008].
- [28] E. Jendrock, J. Ball, I. Evans, S. Fordin, and K. Haase, *The Java EE 5 Tutorial*, Addison-Wesley, 2006.
- [29] "JSON" Available at: <http://www.json.org/> [Accessed March 29, 2009].
- [30] "Method in text-analysis: An introduction" Available at: <http://www.cch.kcl.ac.uk/legacy/teaching/av1000/textanalysis/method.html> [Accessed May 26, 2008].
- [31] "Jena Semantic Web Framework" Available at: <http://jena.sourceforge.net/> [Accessed July 21, 2008].
- [32] "Hook Method" Available at: <http://c2.com/cgi/wiki?HookMethod> [Accessed March 22, 2009].
- [33] M. Melenhorst and M. van Setten, "Usefulness of Tags in Providing Access to Large Information Systems," *Professional Communication Conference, 2007. IPCC 2007. IEEE International*, IEEE, 2007, pp. 1-9.

- [34] “SAP NetWeaver: Components & Tools” Available at: <http://www.sap.com/usa/platform/netweaver/components/index.epx> [Accessed December 15, 2008].
- [35] “glassfish: GlassFish - Open Source Application Server” Available at: <https://glassfish.dev.java.net/> [Accessed December 15, 2008].
- [36] “Welcome to NetBeans” Available at: <http://www.netbeans.org/> [Accessed December 15, 2008].
- [37] “Eclipse.org home” Available at: <http://www.eclipse.org/> [Accessed December 16, 2008].
- [38] “subversion.tigris.org” Available at: <http://subversion.tigris.org/> [Accessed December 16, 2008].
- [39] “subclipse.tigris.org” Available at: <http://subclipse.tigris.org/> [Accessed December 16, 2008].
- [40] “SAP Developer Network (SDN): Downloads, Discussions, eLearning, and Documentation for Developers” Available at: <https://www.sdn.sap.com/irj/sdn> [Accessed December 16, 2008].
- [41] “Java EE at a Glance” Available at: <http://java.sun.com/javaee/> [Accessed December 16, 2008].
- [42] “JA400 Introduction to Java EE 5 SAP NetWeaver,” 2005.
- [43] “JSF Tag Reference” Available at: <http://www.jsftoolbox.com/documentation/help/12-TagReference/index.jsf> [Accessed December 18, 2008].
- [44] “The BalusC Code: Using datatables” Available at: <http://balusc.blogspot.com/2006/06/using-datables.html> [Accessed December 16, 2008].
- [45] “Yahoo! Search Web Services - YDN” Available at: <http://developer.yahoo.com/search/> [Accessed December 18, 2008].
- [46] “Java Developer Center - YDN” Available at: <http://developer.yahoo.com/java/> [Accessed December 18, 2008].
- [47] “JavaServer Faces Technology” Available at: <http://java.sun.com/javaee/jaserverfaces/> [Accessed December 3, 2008].
- [48] “Answers.com - Online Dictionary, Encyclopedia and much more” Available at: <http://www.answers.com/> [Accessed November 25, 2008].
- [49] “Networking Properties” Available at: <http://java.sun.com/javase/6/docs/technotes/guides/net/properties.html> [Accessed November 25, 2008].
- [50] “RFID Smart Shelf & Smart Rail - Checkpoint Europe - EU” Available at: <http://www.checkpointeurope.com/app/?topsectionid=11004&topsection=productgroup§ionid=§ion=&subsectionid=&subsection=&bottomsectionid=&bottomsection=&locale=eu&page=groupproduct&id=1762> [Accessed November 18, 2008].
- [51] “SmartMeter - Product - SM001 Encoded / Pulse N1.0m3/h” Available at: http://www.smartmeter.co.uk/product.php?prod_id=1 [Accessed November 18, 2008].
- [52] “SunSPOTWorld - Project Sun SPOT Products” Available at: <http://www.sunspotworld.com/products/> [Accessed November 27, 2008].

References

- [53] “Wikipedia:Size of Wikipedia - Wikipedia, the free encyclopedia” Available at: http://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia [Accessed December 1, 2008].
- [54] “<https://forge.sap.corp/socrades>” Available at: <https://forge.sap.corp/socrades>.
- [55] “Wink - [Homepage]” Available at: <http://www.debugmode.com/wink/> [Accessed March 25, 2009].

Declaration

UNIVERSITÉ DE FRIBOURG SUISSE
UNIVERSITÄT FREIBURG SCHWEIZ

FACULTE DES SCIENCES ECONOMIQUES ET
SOCIALES / WIRTSCHAFTS- UND
SOZIALWISSENSCHAFTLICHE FAKULTÄT



DECLARATION

Je, soussigné(e), déclare sur mon honneur, que j'ai personnellement préparé le travail qui précède et que celui-ci est conforme aux normes de l'honnêteté scientifique.

J'ai pris connaissance de la décision du Conseil de Faculté du 09.11.2004 l'autorisant à me retirer le titre conféré sur la base du présent travail dans le cas où ma déclaration ne correspondrait pas à la vérité.

....., le20.....

.....

(signature)