Technische Universität Dresden

Department of Computer Science
Chair for Computer Networks
Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

MASTER THESIS

Retrieval of User Interface Templates Based on Tasks

Paoli Izquierdo Llanes

born on October 28, 1980 in Guadalajara, Mexico

Supervisor: M. Sc. Jordan Janeiro

January 15, 2010.

Acknowledgements

I would like to thank my advisor, M. Sc. Jordan Janeiro for providing me with guidance and motivation throughout the duration of this work. I also thank the National Council on Science and Technology from Mexico (CONACYT) for their support during the entire duration of my Master's studies. Last but not least, I want to thank my family and friends for all the encouragement and understanding they have provided for me.

Contents

List of Figures

Listings

List of Abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheet
ELKB	Electronic Lexical Knowledge Base
ILI	Inter-Lingual-Index
ILIR	Inter-Lingual-Index Record
MVC	Model/View/Controller
NLP	Natural Language Processing
OWL	Web Ontology Language
RDF	Resource Description Framework
SWD	Semantic Web Document
SWDB	Semantic Web Document Database
SWO	Semantic Web Ontology
UI	User Interface
URI	Uniform Resource Identifier

1 Introduction

According to the *Survey on user interface programming* conducted by Myers and Rosson[MR92], the time spent on the user interface portion of an application is in average 45% during the design phase, 50% during implementation and up to 37% during maintenance. This represents nearly half of the time of the development of an application as a whole. If the user interface programming and design did not correspond to such a time intensive task, the delivery time of an application could be much shorter and more time could be dedicated to the functionality and quality assurance portions.

This issue also affects the development of web applications and web pages in general. An approach to reducing the time spent on user interface creation in this specific field is by assembling a repository of web templates that can be reused over time. In this way a developer can procure a template from the repository and make only small modifications, if any, to adapt it to the specific application that is being developed or, in the best-case scenario, reuse the template as it is.

Ideally, a web template repository will be filled with several items that vary in different aspects, such as, the tasks it allows to execute, the theme it has been designed for, the context to be used under and the target device. Having a large and rich repository can give application developers the ability to fulfill most user interface requirements.

The problem that arises with this solution, however, is that the task of finding a suitable web template can also be highly time-consuming when the number of available options is too large. Therefore, to effectively exploit the benefits of reusing web templates, the annotation of the characteristics of the templates and an effective search mechanism are necessary.

A useful web template will be the one that will allow a certain number of tasks to be performed in it given a specific application. It is because of this fact that special emphasis must be put into the annotation of the task-capabilities of each of the templates in a repository. Considering that, within a group of people, a task could be phrased in different ways, synonyms are to be considered for the task descriptions when annotating a web template to increase the precision of the results returned by the system.

1.1 Objectives

A web template repository allows a community of application developers to collect user interfaces, which can meet very diverse requirements. As the repository is first created, there may possibly not be too many templates available and the users can easily browse through the few options available in order to find the right one. As new items are added to the repository, its capability of providing more user interface options increases, thus raising the probabilities of successfully finding a suitable template regardless of the functionality required. The drawback to a large repository however, is that, browsing to find a template turns into a more time-consuming chore.

In order to shorten the time necessary to and simplify the process of finding the required user interface, each one of the templates is annotated with a series of attributes, the most important of which is the description of the tasks it is capable of accomplishing. With web template classifications and task descriptions in place, a search mechanism retrieves the user interfaces according to specific user inquiries.

The following subsections will outline the objectives intended for this approach and present the use case for the system.

1.1.1 Web Template Classification

A web page or a web template can be classified based on different aspects. According to Qi and Davidson[QD09], some these aspects include but are not limited to categorization by subject, by function and by sentiment. The diverse currently existent classifications have emerged in order to satisfy specific requirements. For example, when the goal is to create a simple directory based on web page content, a simple classification based on the feature of theme is sufficient. For our approach, however, a simple classification by theme for which a template is designed is not sufficient because there could be many templates in a repository for the same theme that differ in other aspects, such as the device it is designed for, the type of context it is intended to be used in or the functionality it can provide. As a result, a search mechanism could return several results for the same theme and still make it necessary for the user to browse through a large selection of templates to find the right one.

Consequently in order to provide an accurate definition of a web template, the classification will be based on the following aspects:

- Theme
- Context
- Title

- Device
- Structure
- Task

1.1.2 Tasks and Synonyms

The functionality a web template is capable of performing is directly related to the functionality of an application, therefore, this is the most important feature that application developers will consider when searching for templates. Hence, the description of tasks should have enough detail in order for the users to be able to find the correct web templates according to their needs. In addition and in order to further detail the description of the functionality that a web template can perform, synonyms should be used in order to provide for alternative ways to find templates.

1.1.3 Search Mechanism

The process of finding the correct web template in a large repository should be quick and precise and a search mechanism is the tool that can accomplish both of these goals.

The annotations described in the previous sections allow for a search to be performed by taking any and all of the annotation criteria into consideration. While the perfect case scenario would be that the search mechanism should always be able to find the exact match of a query of the user, it is also possible that a web template that meets all the requirements of the user may not exist in the repository. This is why it is necessary for the search mechanism to be able to offer alternatives that meet only certain criteria but only in the cases where no exact matches exist, so that the user can at least modify an existing template to adapt it to the requirements of the application instead of having to go through the process of creating one completely. The opposite situation is also possible, that is, that performing a search could return several results that meet all criteria. In this case, the search algorithm should be able to distinguish which one of the results is more accurate than the next. In order to achieve this, a ranking of the results is necessary.

1.1.4 Use Case

A scenario in which this system could be useful is the following: in a software development company that focuses on creating web applications, a team of developers works on a new application to allow users to search for coffee shops based on the search criteria that the users enter. The application should display a map with the location of the coffee shops found in the area and their exact addresses should also be listed. The team creates a

web service that connects to the database that stores the information about the coffee shops and then they design the web page that will display the information retrieved from the database.

After finishing the coffee shop locator project, the developers get a request for an application with similar functionality but a different theme. For a company like this, the approach of having a repository that stores annotated web templates would be really helpful because they create applications that have similar functionality; for example, for two applications that require a search function, a map display and an address list, the same template could be used. In addition, with a web template repository, they could have searched for a web template with this functionality and, when they were developing an application for a different theme, they could have found a template with the required functionality and they may have possibly just needed to change the background and the images, in order to change from something related to coffee shops to any other theme. Thus, by making only small changes to existing web templates, they could save a lot of time in the development of the user interface of their applications.

In a large company it is possible that several development teams work on different projects and are not aware of the details of all the applications being developed. It is also possible that in a large company with several projects, a large number of web templates will be gathered and that the developers will use different vocabulary to describe the same task. For example, a group of developers creates a web page for displaying music videos and, later on, another development team creates a web page for displaying short art movies. Essentially, the web pages execute the same functionality, which is playing a video file. However, the former team describes the task of the template as '*play video*', and the latter describes the task of their template as '*reproduce movie*'.

In this case, if the users, at the time they annotate the web templates also assign synonyms, they could associate '*reproduce movie*' as a synonym to '*play video*' and thus, the template could be discovered by either of the descriptions.

To summarize, collecting web templates in a repository provides the capability of having a centralized place to retrieve them from, while annotations allow them to be identified by their features. Furthermore, synonyms increases the probability to discover web templates, even if their common functionality is described using different vocabulary.

To begin with the subject matter, *Background Knowledge* (Chapter 2) presents a definition of web templates and tasks and makes a comparison of existing lexical databases. Chapter 3, *Related Work*, gives an outline of similar approaches of search algorithms, web template classification and task taxonomies. In Chapter 4, *Template Model*, the task model is explained in detail with the utilization of user-defined tasks and synonyms to further and more accurately describe the tasks. The *Web Template Retrieval* approach is illustrated in detail in Chapter 5. The *Implementation* details are presented in Chapter 6. The

Evaluation strategy is presented in Chapter 7. And, finally, this work is summarized in Chapter 8 and the *Conclusions* drawn are presented.

2 Background Knowledge

As motivated in Chapter 1, user interface (UI) design and development is time-costly to the overall software development cycle, it is for this reason that UI reuse is encouraged. Particularly in the area of web application development, web templates offer a good solution, however, not every single web template can prove useful to any web application because each application has different requirements in terms of appearance and functionality. Therefore, in order to be able to identify exactly what it is that can be accomplished by the use of a given web template, its functionality must be accurately described.

In Section 2.1 a definition of web templates is offered to clarify how it is that the templates help in the reuse of graphical user interfaces in the area of web applications. Section 2.2 offers an overview of how this work proposes to represent the functionality a web template supplies. Finally, Section 2.3 presents examples of lexical databases that offer the ability to retrieve synonyms.

2.1 Web Templates

According to[Gra09], a web template is a visual model that encloses all the aspects that have to do with the appearance of a web page. These aspects include the color scheme, font style and size, background images, layout and widgets. In technical terms, a web template typically includes not only an HTML file, but also a Cascading Style Sheet (CSS), supporting image files and, in certain cases, it may also contain multimedia files, such as, audio and video.

One of the main reasons to use web templates is to separate business logic from presentation logic[Cor03]. The separation of these two aspects is useful because it allows developers to use the better-suited tools for each one of them since normally the business logic and the presentation logic will use different programming languages. Another advantage is that this separation allows for the development of the two parts to occur in parallel and, in that way, to assign specialized human resources in each area. Another alternative is to have the development workforce to focus entirely on the business logic while having the ability to purchase web templates by third parties. Finally, by separating concerns in this manner, the readability of the code is better because the presentation

code is in one place without having the intrusion of the business logic code and vice versa.

One example in which separation of concerns is used is the Model/View/Controller (MVC) Pattern[GHJV94]. In this design approach, the model is responsible for the business logic of the application, while the view is the graphical representation of the model. Finally the controller is in charge of handling input and calling on the appropriate model object to produce the correct response.

For instance, a product inventory application on the web is a good example of how the MVC pattern can be applied (Figure 2.1). The model would be composed of a '*Product*' class, which would have the attributes of '*Name*', '*Description*', '*Price*' and '*Quantity*'. The view would consist of a web template, which, additionally to all the presentation and styling aspects, would have text areas to display the details of a product. Finally, the controller will be in charge of taking the user input from the view and relaying it to the model. This way, for example, when a user updates the quantity of a product through the view, the controller will take the product data and update the state of the model, which would proceed to update the web template accordingly.

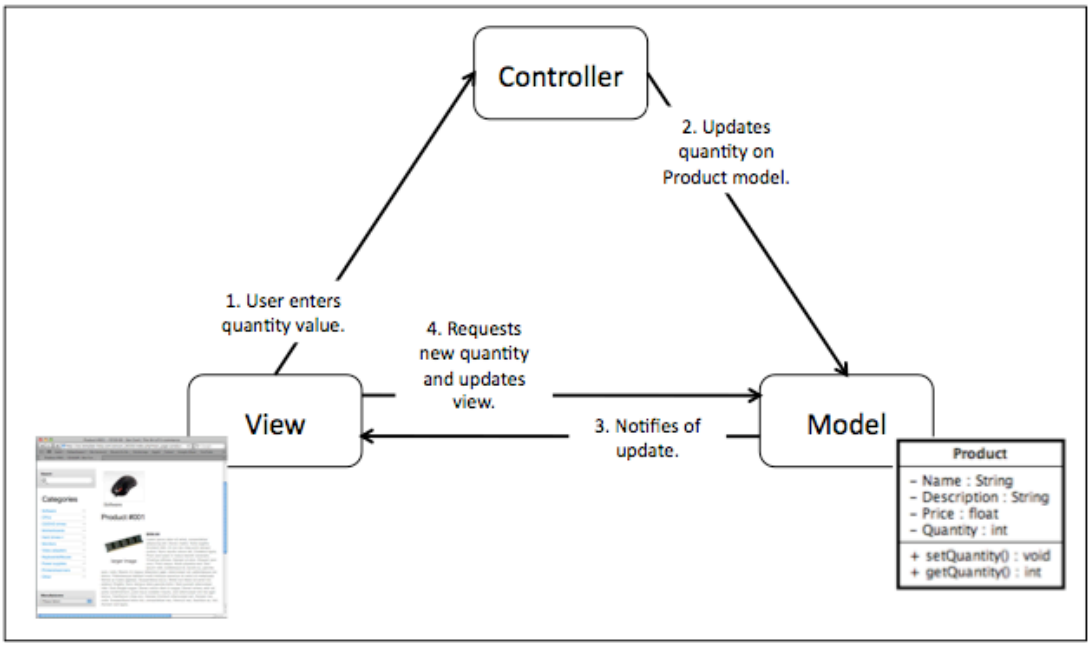

Figure 2.1: Model-View-Controller pattern.

By separating concerns, a web template is only loosely coupled in an implementation. This allows for it to be reused on numerous applications with similar graphical user

interface requirements because the template does not enclose business logic that would interfere when attempting to make use of it on diverse development projects.

2.2 Tasks

According to[DS04] a task is the procedure through which one or several entities, that include humans or computers, change an application domain. For example, in a money transfer transaction on an online banking application, the entities involved are the holder of the bank account and the web application, because the account holder is responsible for giving the order to transfer the funds from one account to another, and the web application is the one that establishes a connection with the bank and carries out the transaction. The actual tasks in this scenario are giving the order for the transfer and executing it in the database of the bank. Specifically in this work, a task refers to the functionality that can be accomplished by using a web template.

Annotating a web template with the tasks it is able to perform, will make it possible to distinguish it from others and narrow down the search for a web template that meets the requirements of a given search query. This is why the tasks a template is capable of performing must be described as thoroughly and precisely as possible. Two of the methods found that can be used to achieve this goal are through a controlled vocabulary or by using free text.

In the technical report *The ERCIM Technical Reference Digital Library* by Andreoni[ABB+99] et al., the disadvantages to each method are pointed out.

> "Controlled languages evolve slowly, so that their terms may not be capable of representing specific document contents nor user information needs (particularly true for those disciplines where new terms are continuously coined)."

Computer science is clearly one of those areas that continuously create new terminology to be able to describe advances and breakthroughs in the field. In consequence, this could present a difficulty in finding a controlled language that could efficiently describe web templates.

Andreoni[ABB+99] et al., continue with the disadvantages on free text.

> "The inverse is true for indexing with coordinated keywords selected by the user. Such keywords represent document contents and also avoid false coordination; however, they suffer the same drawbacks as free word indexing with respect to the probability of successful matching."

Allowing users to express the functionality of a web template using vocabulary that is familiar to them helps against the slow evolution of controlled vocabularies. Nonetheless,

performing a search based on free text may not yield very accurate results because of the different ways the users can express these descriptions in terms of vocabulary, grammar and writing style.

Combining both of these methods, however, can reduce the limitations each one of them presents single-handedly. Hence, a task model based on a task taxonomy is adopted as a way to uniformly annotate web template functionality. At the same time and in order to increase the preciseness of the representation of a task in the annotations, the users of the system should be able to enter their own task descriptions as free text, which could subsequently, allow them easier access to the template in the repository. Having the users describing web templates using their own vocabulary means that the descriptions stored will have the words most likely to be used later on when someone searches for a template.

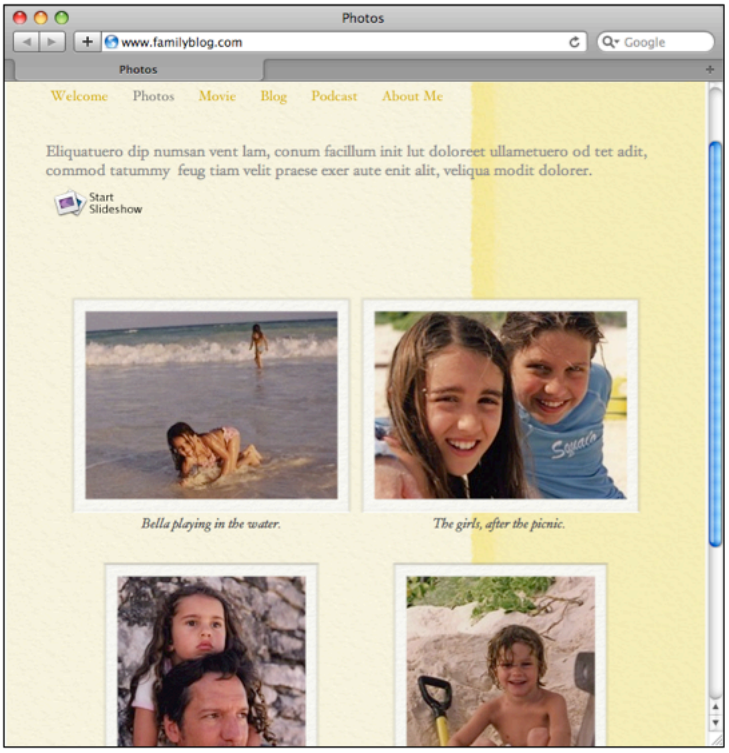

Figure 2.2: Many different terms can be used to describe a web template.

One more factor to consider in user task descriptions is the fact that not every person will necessarily use the same vocabulary when expressing the same idea. For example, in a survey conducted for this work where a group of people was asked to give an open description of a set of web templates, six different terms were used in order to describe the use of photographs in the template seen in Figure 2.2, that is, '*picture preview*',

'*photos*', '*photo albums*', '*thumbnails*', '*photo book-like layout*' and '*slideshow*'. This is why it is important to allow not just one, but rather any and all users of the system to enter their own user descriptions to whichever of the templates are available in the repository. To further aid in not only the task description but on the search as well, synonyms should be used to complement and enhance the user task descriptions that are assigned to a web template because the synonyms will provide different ways in which a user can express the descriptions of a task.

2.3 Lexical Databases

The process of finding the synonyms to a specified word can be achieved through the use of lexical databases and in order to find the better solution for our approach, three lexical databases were evaluated: WordNet, EuroWordNet and the Electronic Lexical Knowledge Base of the Roget's Thesaurus. These solutions are presented in the following subsections.

2.3.1 WordNet

One of the most widely used lexical databases for the English language is WordNet[Mil95].

WordNet offers an online lexical database tool where each word belongs to a syntactical category, or part of speech, that reveals whether a word is a verb, noun, adjective or adverb. In order to determine the synonym sets (or synsets), each word has a sense, which determines its connotation and is extracted from a collection of meanings. Thus when two or more words share a sense they belong to the same synonym set. Furthermore, this lexical database is able to produce results for a verb even when the query posed for it is not in its infinitive form, i.e. a query for the word '*wove*' returns the existing information on the word '*weave*', because inflectional morphology is accommodated for all syntactic categories. However compound words and expressions derivative from other terms, like '*guide*', '*misguide*', '*guidable*', are considered as distinct from one another.

In addition, WordNet incorporates a set of semantic relations between words and word senses. They are the following:

- *Synonymy.* Words with this relationship share a word sense and are therefore deemed as synonyms.
- *Antonymy.* This relationship is the opposite of the previous one and denotes words with differing meanings.
- *Hyponymy / Hypernymy.* These relationships create a hierarchical structure where the top level of the hierarchy is the super-name (hypernym) and its children are

sub-names (hyponyms). For example, '*animal*' is a hypernym whereas '*mammal*' is a hyponym.

- *Meronymy / Holonymy.* These concepts describe the relationship between part-names (meronyms) and whole-names (holonyms). For example, a '*soldier*' is a meronym and '*platoon*' is its holonym.
- *Troponomy.* This is a similar relationship to hyponymy but applied to verbs instead of nouns.
- *Entailment.* This is a relationship for verbs where one of them is necessary for the other to occur. For example, for a person to graduate it is necessary for him or her to have studied some discipline, therefore the verbs '*graduate*' and '*study*' share an entailment relationship as the latter is required for the former to ensue.

WordNet can be accessed via an online service where the results to a query display the synonym sets for each sense a word has, grouped by part of speech. Additionally a glossary is also included so as to clarify the meaning of that collection of words, which may or may not additionally present a sample sentence. To enable the use of this lexical database outside of the web, the developer community has created several Application Programming Interfaces (APIs) than can be used under different programming languages, such as Ruby[1] and Java[2] among numerous others[3].

2.3.2 EuroWordNet

Building on the concept of WordNet, the European Commission funded the EuroWordNet [Vos98] project, which created multiple wordnets for different languages spoken in Europe in order to generate a multilingual lexical database. Initially only Dutch, Italian, Spanish and English were the languages contemplated in the project, with wordnets of smaller dimension (circa 30,000 synsets per language) than the original English language project by the Princeton University. However four more languages, German, French, Estonian and Czech, were later on added to the project covering even fewer synsets (between 7,500 and 15,000 synsets).

The data structure of EuroWordNet is made up of wordnets, whose database structure is based on version 1.5 of WordNet, for each of the languages previously mentioned but that are restricted to nouns and verbs. In order to denote equivalence among synonym sets of different languages, the Inter-Lingual-Index (ILI) is included as part of the architecture. The ILI holds records, each of them containing a synset, an English gloss and a reference

[1] http://www.deveiate.org/projects/Ruby-WordNet/
[2] http://wordnet.princeton.edu/wordnet/related-projects/#Java
[3] A complete list of the local Application Programming Interfaces for WordNet is found in http://wordnet.princeton.edu/wordnet/related-projects/#Java

to its source. Synonym sets from the language specific wordnets that link to the same ILI-record (ILIR) are therefore considered equivalent. At the same time, ILIRs do not uphold any relationships with other ILI-records. The other two entities that can create relationships with the ILI are the Top Ontology and the Domain Ontology.

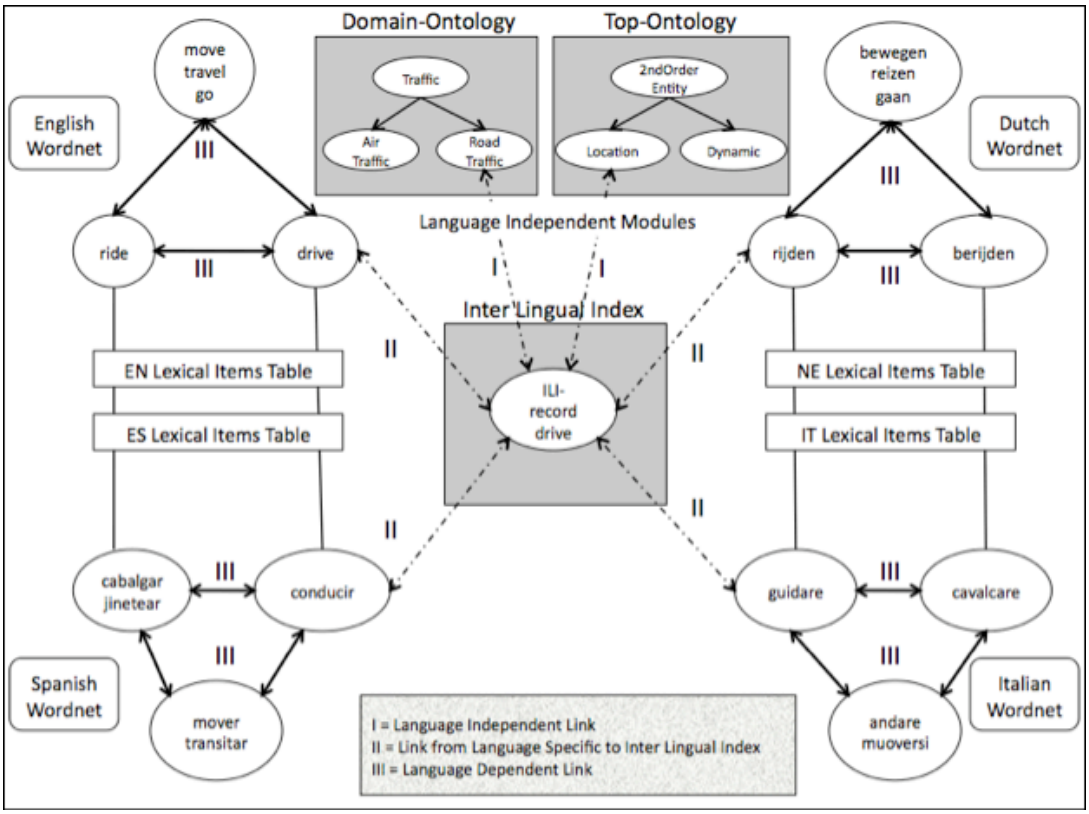

Figure 2.3: EuroWordNet Data Structure Architecture[Vos98]

A group of the most important concepts in the different wordnets comprise the Top Ontology so that uniform classifications can be made across the different languages. Similarly to synonym sets, a concept within the Top Ontology also holds associations to ILIRs, for instance, as seen in Figure 2.3, the concept of '*Location*' holds a link to the ILIR '*drive*', which in turn is associated to the words '*drive*', '*rijden*', '*guidare*' and '*conducir*', meaning that these terms are also connected to the top concept of '*Location*'. Furthermore, given the internal relationships of the words within their own language, the top concept is once more inherited.

The domain labels contained in the Domain Ontology are inherited to the language-specific words in the same manner as the top concepts. The purpose of the Domain

Ontology is to enable the grouping of concepts by script as opposed to by classification, as well as to give the capability to discern generic from domain-specific vocabularies.

2.3.3 Roget's Thesaurus

An approach to manually find synonyms is provided by thesauri and the Roget's Thesaurus [Rog91] supplies an exceptionally broad list of general-purpose lexicon. As pointed out by [KS08] and [JS03], the organization of the thesaurus is in the form of a hierarchy of nine levels. The first level is formed by eight classes: *Abstract relations, Space* and *Matter*, which refer to the external world; and the rest, *Formation of ideas, Communication of ideas, Individual volition, Social volition* and *Emotion, religion and morality*, that deal with the inner world of humans. Parting from those classes, the hierarchy is followed by 39 sections, 79 subsections, then 596 head groups and 990 heads distinguished by numbers. Consequently, 3220 part-of-speech groups make up the sixth level that comprises the noun, verb, adjective, adverb and interjection categories. Subsequently 6443 paragraphs follow, which are in turn formed by a total of 59,915 semicolon groups. The final level is the one composed by words that, at a total of 225,124, are part of a synonym set when they belong to the same semicolon group. To elucidate the classification explained, the thesaurus entry for 'Place', retrieved from the 1911 edition [Rog91] of the book made accessible by the Project Gutenberg[4], is shown below.

> "Class II Words relating to space
> Section I. Space in general
> 1. Abstract space #182. [*Limited space.*] Place - - N. place, lieu, spot, point, dot; niche, nook &c. (*corner*) 244; hole; pigeonhole &c. (*receptacle*) 191; compartment; premises, precinct, station; area, courtyard, square; abode &c. 189; locality &c. (*situation*) 183. ins and outs; every hole and corner. Adv. somewhere, in some place, wherever it may be, here and there, in various places, passim."

In the previous example, the class and section are clearly marked and are followed by the subsection '*Space in general*', the head group '*Abstract space*' and the head number 182 for the word '*Place*'. Subsequently the head is divided into the parts of speech Noun ('*N.*') and Adverb ('*Adv.*'), with one paragraph each, and nine semicolon groups for the Noun part of speech and only one for the Adverb.

Given the potential for usability this thesaurus presents for the field of Natural Language Processing (NLP) Jarmasz and Szpakowicz[JS01] produced an Electronic Lexical Knowledge Base (ELKB) version out of the edition from 1987 and a Java application to access it. One drawback that the Roget's Thesaurus presents though, is the lack of an

[4] http://www.gutenberg.org/ebooks/22

explicit representation of semantic relationships among words. In view of the fact that WordNet does clearly provide this feature, they were able to assign semantic relationships to certain semicolon groups when, through a comparison against a WordNet synset, both structures had at least one word in common. For instance, under this assumption, by performing a comparison of the word '*prototype*' in both lexica, one can find that the hypernymy relationship is established among the words '*model*' and '*example*' within WordNet, hence the semicolon groups '*prototype, original, model, pattern, precedent, standard, ideal, reference, scantling, type;*' and '*protoplast, module, exemplar, example, ensample†, paradigm;*' would be designated as hypernyms to '*prototype*' in the ELKB.

2.3.4 Discussion

Synonyms are used to further enhance the description of a task that is contributed by the user, which makes the need of a synonym lookup service necessary. While all of the approaches discussed in Section 2.3 can effectively provide a series of synonyms, the Electronic Lexical Knowledge Base that uses the 1987 edition of Roget's Thesaurus as a source still has some limitations in the description of the lexical relationships between words. Another drawback this system presents is the fact that computer science-related vocabulary may be necessary to a certain degree in order to describe the tasks of a web page template and this thesaurus is not up-to-date. The solution offered by WordNet and EuroWordNet are therefore more fitting but WordNet is used in our work because it is a more widespread and open system. Additionally, the English language wordnet provided by EuroWordNet is simply an older version of the original WordNet.

3 Related Work

Over the course of the history of the web, there have been many ways in which web pages, and therefore web templates, have been classified. Most of the times the diverse classifications that have been produced are customized in order to respond to a specific problem or area of focus. Classifying a web template based on the tasks it can execute is important, due to the capability of indexing it and retrieving it later.

The tasks that can be generally performed through a web page can also be represented in different ways, so in order to standardize their nomenclature, task taxonomies have been proposed in previous research.

Both the classification of a web template and the descriptions of the tasks it can accomplish play a key role in being able to locate a specific item. There have been a lot of search algorithms developed for different purposes and, in order to achieve precision, it shall have to exploit these metadata to their full extent to efficiently retrieve results.

This chapter covers related work in the area of search algorithms (Section 3.1), web template classification (Section 3.2) and task taxonomies (Section 3.3). As a final point the differences and similarities between these approaches and the present work are presented in Section 3.4.

3.1 Search Mechanisms

Google [BP98], a widely used web search engine, keeps the full HTML of web pages in a repository and indexes it through the use of the MapReduce [DG04] algorithm, which allows for a quick and effective way to sort and count the number of occurrences of a word in a document. In addition, to rank the results found, the PageRank [PBMW98] algorithm analyzes the incoming and outgoing hyperlinks of the documents to determine relevance with respect to the search query terms. As part of a standard web search, the Google approach does not pay specific attention to the metadata of a web page, but it rather evaluates its text and hyperlinks. Nevertheless, there are other projects that are integrating metadata into their search algorithms, such as Google Image Swirl[JR09], still part of Google Labs[1], where the goal is to return sets of images that are similar

[1] http://www.googlelabs.com/

both visually and semantically, thus disambiguating the meaning of a search term. For example, by comparing the visual characteristics of a set of images and their associated metadata, the results for the word '*Zurich*' could be divided into two categories of images, one for images of the city and another for the financial services group.

Structured metadata or semantic information enables computers to understand the contents of a web page without human assistance, which is the purpose of the Semantic Web [Her09]. Currently, most of the information found on the World Wide Web is of the kind that is only understandable by humans, like text, images and video, among others, which make up the content of a typical web page. In [DOS03] Daconta, Orbst and Smith point out that the data used in the semantic web must be "*smart*" in order for machines to understand and process documents effectively. Smart data is the type of information that is not limited to function in a single domain, but rather across several dissimilar ones. Furthermore, data should be inferred by applying a set of rules, thus the use of ontologies [UG96] is suitable in the semantic web to make pages machine processable because they have the ability to represent a collection of concepts, their definitions and associations.

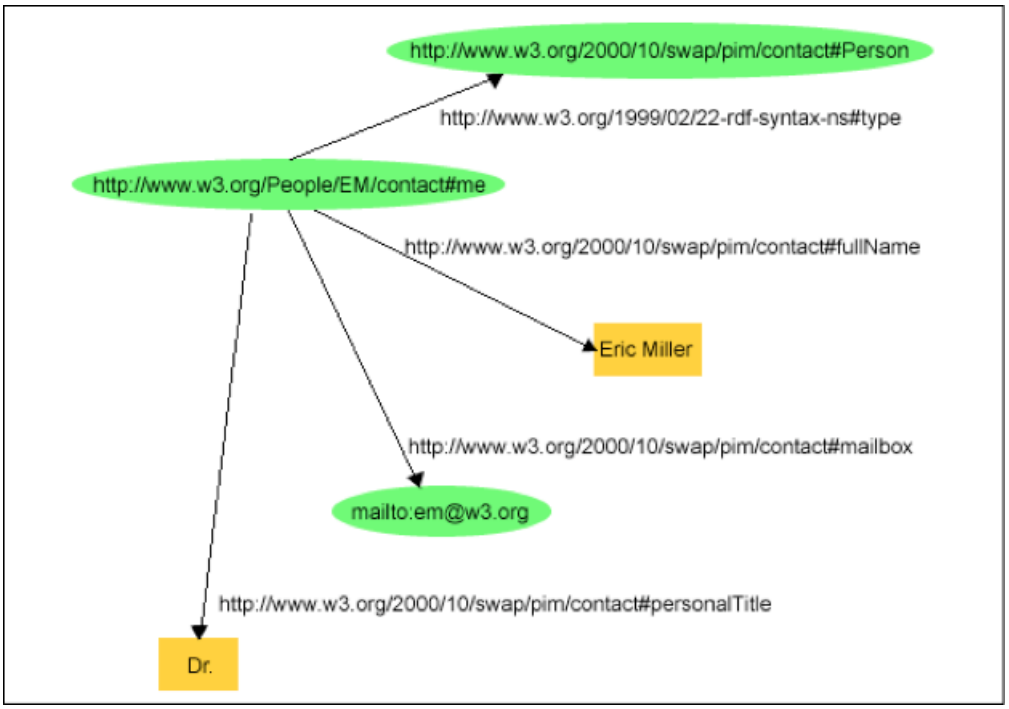

Figure 3.1: RDF graph example.

The technologies used in the semantic web are mainly the Resource Description Framework

(RDF) [Bec04] and the Web Ontology Language (OWL) [MvH04]. RDF is used to represent information or metadata that describes a web page in the World Wide Web in terms of properties and values that are associated to Uniform Resource Identifiers (URIs). RDF data can be represented in a graphical form as seen in Figure 3.1. In this example provided by the W3C Recommendation of RDF [MM04], a depiction is given for the description of the resource of type '*Person*', with the additional properties of '*Full Name*', '*Mailbox*' and '*Personal Title*' and the corresponding values of '*Eric Miller*', '*em@w3.org*' and '*Dr.*'.

The description of resources in this form creates triples composed by a subject, a predicate and an object. Parting from the illustration above, one of the triples can be expressed as "*Person has a full name whose value is Eric Miller*", where '*Person*' is the subject, the '*full name*' property is the predicate and the value of '*Eric Miller*' is the object. Furthermore, in order to enable machines to process the data contained in these triples, an XML-based language can be used for serialization. The previous example is shown here using such convention in Listing 3.1.

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

Listing 3.1: XML representation of RDF graph.

The Web Ontology Language (OWL) [DOS03] adds more functionality to what RDF provides. Even though OWL also uses triples to create the model of an area of knowledge, it also includes further vocabulary to refine the description of the resources and to allow the expression and processing of a first order logic than can be done by computers. As part of this additional vocabulary, OWL does not only establish a relationship between two classes, but details it by stating, for example, whether they are disjoint or complement each other. In terms of equality OWL provides with the capability to specify whether resources are the same as one another or equivalent of each other. Furthermore, relationship properties can also point out whether they are symmetric, transitive or the inverse of each other and indicate cardinality.

Ontologies are saved to files referred to as Semantic Web Documents (SWDs) and in order to easily find them on the web, search mechanisms are necessary. One example of a search engine developed specifically for this is Swoogle [DFJ+04], which uses a web crawling mechanism to discover SWDs automatically, then caches the results found and extracts their metadata on a semantic and syntactic level and stores it in a database. With the

metadata obtained, an analysis component checks for the ontological relationship between SWDs and ranks them according to those relationships. It also classifies the documents found as either Semantic Web Ontologies (SWO) or as Semantic Web Databases (SWDB). The difference between a SWO and a SWDB lies within the proportion of terms a document newly defines or extends from another SWD, when the proportion is high, then the document is classified as SWO and when it's low, it's deemed a SWDB. The implementation of this system is done through a web interface where users can perform either simple keyword searches or advanced searches based on several constraints, such as words contained on the document URL, range of quantity of triples included and of classes and properties defined, document language, encoding and validation, and percentile of Swoogle rank that the document belongs to.

Another example of a search mechanism for the Semantic Web is presented in [SKV08] where the users employ free-form queries to carry out a search. The system depicted in Figure 3.2, named SAMOS, takes these free-form queries and uses WordNet senses in order to disambiguate the meaning of the words. It then generates triples that consist of concepts and relationships between these words thus generating an ontology. At the same time, it uses the terms entered as a free-form query on Swoogle to retrieve SWDs. The SWDs found are then matched against the ontology created by the term disambiguation phase and a relevance score is calculated based on the number of mapped concepts.

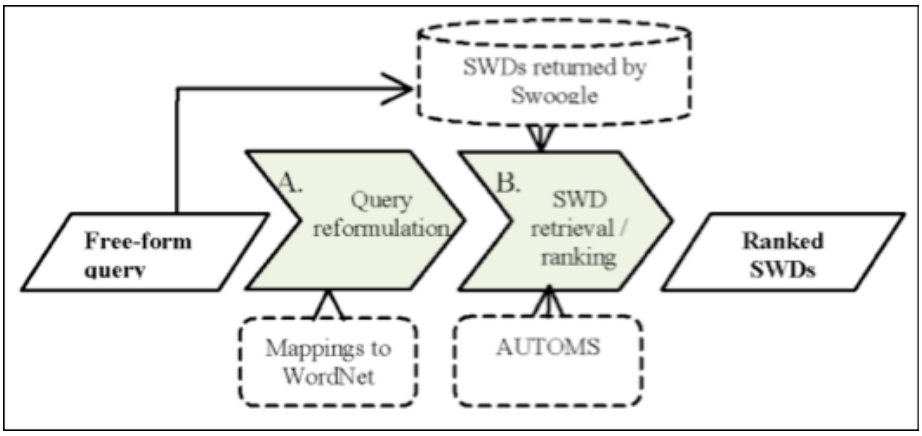

Figure 3.2: SAMOS Architecture.

On the other hand, [DEFS06] offers a different approach to the annotation and search of annotated web documents. With the objective of improving the quality of searches on corporate intranets, rather than on the Internet, this project focuses on adding annotations to intranet pages both explicitly and implicitly. The explicit form, which is similar to Google Sidewiki[2], entails having the users of the searching mechanism add

[2]http://www.google.com/sidewiki

annotations that describe the pages they visit on the corporate intranet. The implicit annotations are added to a page by means of the search terms a user entered to find that page. Both explicit and implicit annotations are saved to an annotations database and included in the search engine index; however, they are treated with no more or less relevance than anchor text during searches.

3.2 Web Template Classification

The approaches taken to classify web pages or web templates can be very different from one to another; this is mainly due to the fact that the classification is usually created to suit a specific objective. For example, some classifications may be based on the geographical location a page belongs to, others by the theme of their content and others simply by the top-level domain of the page.

In [GPT05] Gibson, Punera and Tomkins analyze the amount of templated material on the web, which they define as *"common content or formatting that appears on multiple pages of a site"*, and its behavior. As part of their research they divide web pages into the following seven categories without considering their content: Brochure, Catalog, Community, Documents, News, Personal and Portal. '*Brochures*' are corporate web sites that include information about the history, background, news and other details pertaining to the organization; '*Catalogs*' on the other hand display a series of products; the '*Community*' category refers to sites where its contributors are several different people, like what is commonly known as forums; '*Document*' refers to sites that include citation material like scholar sites; '*News*' deals with pages that have content made up of several reports in one or multiple subject matter areas; the '*Personal*' classification encompasses pages belonging to individuals, which can have a mixture of different types of content; and, finally, pages made up of links to other pages are categorized as '*Portal*'.

Lindemann and Littig in [LL06] perform a study where they make functional classifications of web pages and, based on their structural properties, they examine the relation between the function of a website and its structure. The functional classifications established here were Academic, Blog, Corporate, Personal and Shop, which they believe to be categories where most web pages can be allocated. The structural properties include the size of a web site, its organization, how its URL is composed and, finally, its link structure. These factors are measured by the number of pages of a site, the average number of slashes and digits in the URL path, the average URL and URL path length, the length of the name of a site, the number of sub domains, the fraction of document types, the average and maximum outdegree, the average and maximum internal and external outdegree, the average and maximum external site outdegree, the number of languages found in the site and the average document size. For classification purposes, they use a naïve Bayesian classifier function and, ultimately, with the intent of improving the accuracy of

the process, they use a group of probability density functions, which include exponential, normal, longnormal, Weibull and Pareto. Their findings revealed, among others, that sites classified as '*Academic*' mainly stand out from those under other categories by their size, while '*Shop*' sites differentiate themselves on account of their average internal outdegree.

In a similar approach, [MF99] Matsuda and Fukushima propose a classification of web pages according to their structural characteristics. By tagging a web page with the parameters of keywords, links, URL, structure, image, OCR and plug-in they are able to find the problem-solving task that the web page is able to perform. For example, according to their research, a product catalogue web page has a very high likelihood of having the keywords '*product*', '*specification*' and '*feature*'. If, in addition, the top-level domain found in the URL is .com and the word '*product*' is also part of the path, the reliability for this being a product catalogue web page increases.

A different classification method is used in a study to determine the success of a website[SFB06], in which they are divided by their specific goals into the categories of Informed decision – biased, Informed decision – unbiased, Life enrichment, Online learning, Entertainment, Knowledge enhancement, E-Commerce, Online Community, Information specific search, Interactive service management and Online application. According to this categorization, '*Informed decision – biased*' sites are those that attempt to persuade the purchasing decision process of their clients, whereas the '*Informed decision – unbiased*' does not present this favoritism. On the other hand, the '*Life enrichment*' class refers to a topic instead of a product. '*Online learning*' has an aim to provide a didactic environment, while '*Knowledge enhancement*' only offers brief updates or focuses on a single subject matter area. Furthermore '*Entertainment*' sites contain games and music among others and sites where online transactions take place fall under '*E-Commerce*'. The '*Online community*' category refers to forums where people can collect and contribute with information regarding a specific issue. Moreover, for '*Information specific search*' the sites considered are search engines. Ultimately '*Interactive service management*' sites allow their users to manage accounts of some sort and '*Online application*' sites are those that enable access to a web-based program. After conducting the survey on several subjects, specific models were suggested as to what measures are to be contemplated in order for a site to be successful based on the category it belongs to. For example, while system quality should be considered for web search engines, it is not relevant for online forums, for which social relevance is a more significant aspect in their success.

3.3 Task Taxonomies

Task taxonomies [HS84] enable the classification of activities that can be performed in a discipline. Specifically for our work they can allow us to create a hierarchical method to

represent the tasks that can be accomplished through a web template, as well as provide a fixed model that may encompass the majority of tasks most commonly carried out through a variety of them.

Byrne et al., during a study of Internet usage [BJWC99], followed the activities of a group of people while browsing the Internet and constructed the task taxonomy seen in Figure 3.3. They did not only base this taxonomy on the findings of the surveillance of the activities of their sample group, but also on previous research of information-finding, as well as web browser capabilities and personal introspection. The web tasks proposed in this taxonomy are:

- *Use information.* This task refers to situations in which the intent of the user is to utilize the content found in a web page. As a result, the subcategories are based on what the user does with that content, hence, read / view / listen, save to disk (download), display for others, duplicate and print are listed.
- *Locate on page.* In order to perform the previous task, the user has to find the information by doing a visual scan of the page. The usual type of data that is searched for are images, specific strings of text or words related to a concept in particular, however there were times when the user also looks for something that appeals to their attention without having a previous intention of locating it. In some other cases users try to find tags that identify what they are looking for when they do not know the exact word to use. The subcategories for each of these cases are image, specific string, related concept, something "interesting" and tagged information respectively.
- *Go to page.* Any activity that ends on a new URL being loaded into the web browser falls into this category. Typically, a user will click on a hyperlink, click on the back/forward button of a browser, select a bookmark, choose an item from the history list or directly provide a URL to cause the web browser to display a different web page.
- *Provide information.* This task encompasses actions where the user supplies some type of information to be sent over the Internet. Some of the most common information users enter on their browsers are search strings, shipping addresses and survey responses, however there can be numerous additional pieces of information than can be provided, therefore the '*etc.*' subcategory is added here.
- *Configure browser.* Configure tasks are those where the user changes the state of the browser, like adding a bookmark, changing the cache size or scrolling and resizing a window. The subcategory '*etc.*' is also used here as there could be further settings changed.
- *React to environment.* When the browser needs user participation, such as, responding to a dialog box or a display change or reloading a page among others,

this task is used to describe those activities.

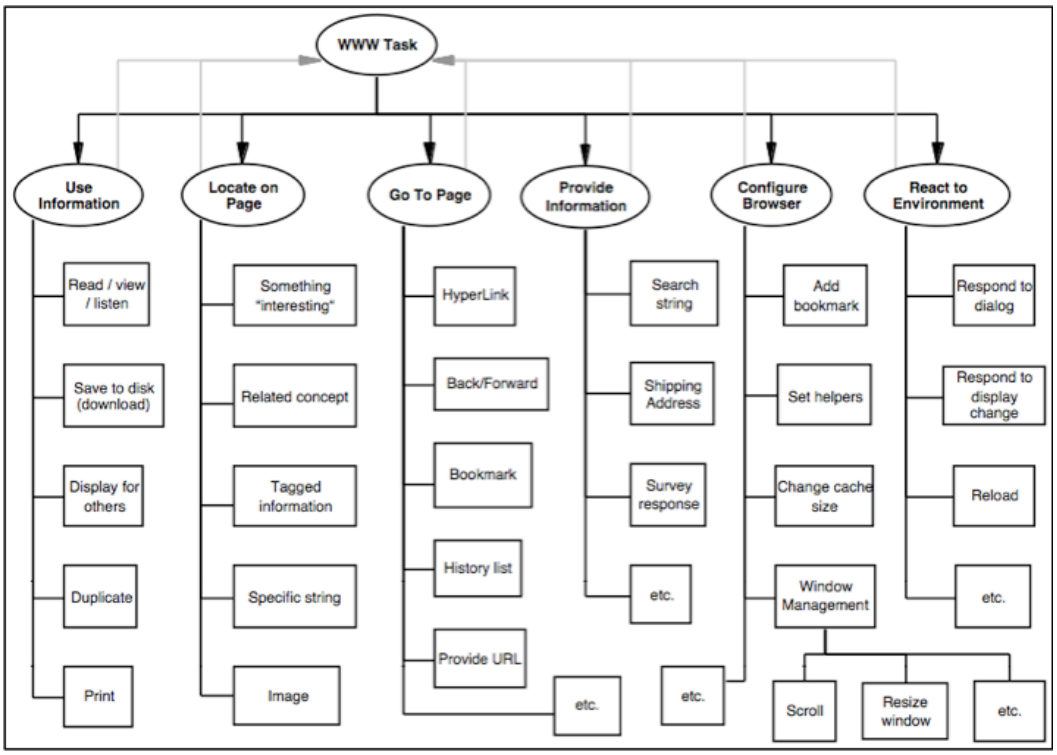

Figure 3.3: Task taxonomy.

Three examples of task taxonomies that describe activity on the web are presented in [MPC01], one based on the reason why users conduct a search on the Internet, purpose taxonomy; another focusing on the manner in which the user finds information on the web, method taxonomy; and one that concentrates on the type of information that is searched for, content taxonomy. These taxonomies were created after evaluating the answers to a survey that asked people to illustrate an instance where they found information on the Internet that conducted them to make a significant action or decision.

The purpose taxonomy is the following:

- *Find.* The purpose is to download data, whether it is a piece of information or a file.
- *Compare / Choose.* This task refers to when the user compares different pieces of information to aid in the process of making a choice.
- *Understand.* The use of information found on the web to comprehend a subject falls into this category.

The components of the method taxonomy are:

- *Explore*. This category describes searches that are not initiated with a specific objective in mind.
- *Monitor*. When a search is part of a habitual process.
- *Find*. When the search does have a particular intention in mind.
- *Collect*. When the goal of the search is to find numerous pieces of information.

The following aspects form the content taxonomy based on what the user searches for:

- Business
- Education
- Finance
- Job search
- Medical
- Miscellaneous
- News
- People
- Product Info and Purchase
  - Computer
  - Vehicles
  - Download
  - Other
- Travel

One further example of a web task related taxonomy is presented in [Bro02] focusing particularly on web searches because the goal of this research is to evaluate the progression of search engines as they intent to meet web-specific requirements. In this approach, the taxonomy is based on the objective of the query performed on a search engine, which can be navigational, informational or transactional. Navigational queries are those where the user wishes to access a specific website and, in this case, there is only one correct result. For example, if the user enters a navigational query for '*Deutsche Bank*', the expected correct result is the homepage for the bank, however the search engine may also list other pages that make reference to it in their content, such as, news pages or blogs. On the contrary, informational queries are those that yield several useful results because the aim is to find information on a particular concept from pages that are not created

dynamically. Ultimately, when the target of a search query are sites where additional interaction takes place, like e-commerce pages or other web service sites, the category assigned is transactional.

3.4 Discussion

A couple of different approaches were presented in Section 3.1 in regards to searching for web pages annotated with metadata. The use of Semantic Web technologies offers a structured method of annotating metadata; furthermore the search mechanisms reviewed here focus the relevance of their results on the ontological relationships between documents. The work presented by [SKV08] is of special interest given that it does not need to rely on existing ontologies to find documents, but creates its own ontologies with the help of the WordNet service. In the approach shown by [DEFS06] the capability to enter annotations in an unstructured manner is permitted and this metadata is consequently added to a search engine index in order to enhance the quality of the results.

After assessing both approaches, in our case, a mixed method of annotations classified in the form of specific key, value pairs that define a web template and its tasks is better suited. Despite the fact that creating an ontology in order to describe web templates can be helpful during the retrieval process, a consensus must be reached among the user community of the system in terms of the vocabulary to employ when annotating the templates. By allowing the user community to annotate the web templates using the terminology of their choice however, the description is customized to fit their own jargon, which ultimately is what they will make use of when they carry out searches. To further enhance these descriptions we propose the usage of synonyms, via the WordNet service, but unlike the approach presented in [SKV08] the user is responsible of choosing the appropriate synonym, if any, to the description they enter to achieve term disambiguation. Moreover, in contrast the approach provided by [DEFS06], we propose a simple structure of key, value pairs to annotate web templates to allow for a more precise search, where the properties (keys) of a template are defined by the system and the users provide their values. In addition, the task of annotating web templates is an ongoing group effort, so the entirety of the users of the system are allowed to annotate any web template and assess the correctness of the descriptions already given to it by others. Thus an assignment of a relevance to each of the different descriptions is possible. Consequently an underlying ontology with a strict vocabulary is not necessary and, at the same time, each annotation has a specific meaning and can be assigned a specific weight during the searching phase.

The approaches presented in Section 3.2 offer diverse ways to classify web templates although most of them establish the classifications in a way that does not define each template with enough specificity to make the process of finding them in a repository more

straightforward. For example, one of the categories suggested by [GPT05] is '*Community*', however, a community could be based on different contexts, for instance, a personal one where people exchange information about a hobby in common, or a community provided by a corporation where their customers share experiences about their products. Web pages with dissimilar contexts such as these would normally also have contrasting designs. Another factor that could help differentiate web templates is the theme they were originally created for and is something that the approaches discussed also do not take into consideration. The approach taken by [MF99], however, does offer a few more details by tagging the web pages with a series of parameters, but the tasks are viewed as just one overall problem-solving duty for the entire the web page, rather than several more detailed tasks, which is the aim of our work.

The description of tasks must be as accurate as possible in order to identify the correct web template according to the requirements of the user. The approach presented by [MPC01] focuses on the tasks that the user sets out to achieve when browsing the Internet, how he or she accomplishes them and the type of content accessed, rather than on the activities that can be executed in a single web page. On the other hand, the taxonomy in [Bro02] centers on the classification of web searches by the type of results that the user expects to find. Finally, while the taxonomy presented by [BJWC99] does present a hierarchy of detailed tasks that one can come across while browsing the web, it also describes tasks that not only concern web templates but the browsers themselves. However, given the fact that this taxonomy does provide a classification that deals with web page templates and their common functionality, a simplified version of it is better suited for this work.

4 Template Model

The description of the web page templates needs to be based on a series of criteria in order to represent its attributes as specifically as possible. If the detailed description regarding the web template is not available then it is more difficult to accurately respond to user inquiries. For example, if web templates were only to be grouped by the device that they are to be used with, we would have a small number of categories each with a large number of elements. Therefore the template model seen in Figure 4.1, which encompasses a series of attributes to describe a web page template, is proposed.

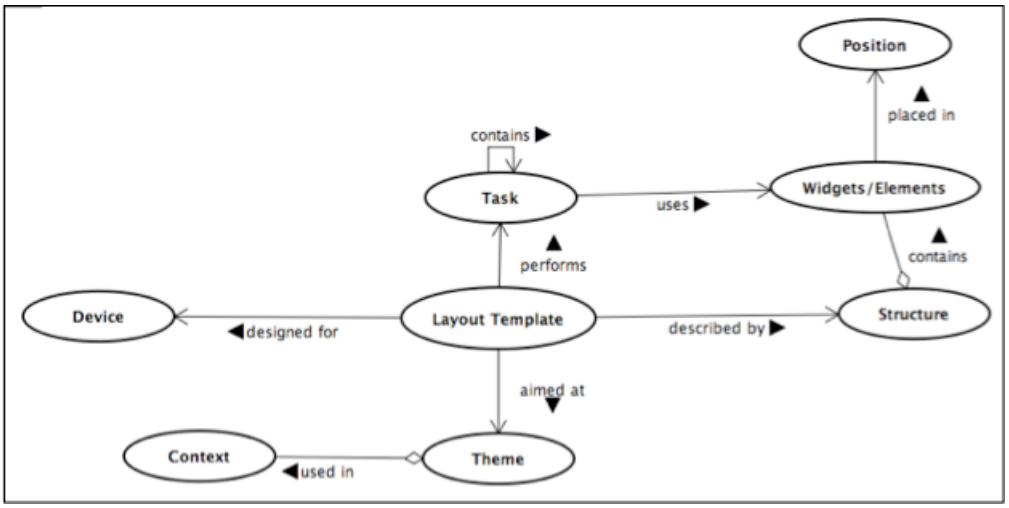

Figure 4.1: Template Model

At the center of the model lies the template itself, which has a structure, which encompasses all its visual elements and how they are distributed across the template. The structure contains a series of widgets that are graphical controls like text boxes, push buttons and checkboxes and they are positioned in specific parts of the template that can be expressed in terms of coordinates. The template performs tasks, which, as presented in Section 2.2, describe the functionality that can be accomplished by using the template. The templates are designed to have a theme that relates to the subject matter of the application they hold and are used in a specific context, which specifies the setting under

which it will be used. Finally, the device specifies the type of tool for which a template is suited to function in.

The functionality a web template can provide is the key reason for the users to search for one in the first place because their primary goal is to find a graphical interface that can be adapted to the functionality of the application they will develop. Therefore, the annotation for tasks requires a higher priority and level of detail, so a task model is proposed and presented in Section 4.1. Also a way for users to formulate tasks with vocabulary that is familiar to them is presented in this section in the form of user-defined tasks and synonyms. Moreover in order to further aid in the web template description process, Section 4.2 gives an overview of all the remaining annotation criteria, that is, theme, context, device, structure, widget/elements and position.

4.1 Task Model

In view of the fact that the primary goal of this system is the reuse of web templates whenever new web services or web applications are to be developed, the descriptions of the tasks that can be accomplished through the use of a template play a key role. Consequently, a taxonomy of the most commonly executed tasks is recommended. As discussed on Section 3.4, the task taxonomy proposed in [BJWC99] is adapted in order to use the tasks that concern more with the web template and eliminate the ones that are irrelevant to it. In addition, other tasks have been added to enhance the taxonomy for use in this approach.

As seen in Figure 4.2, '*Configure Browser*' and '*React to environment*' were the two categories that were removed from the original taxonomy. The reason behind this omission is due to the fact that these activities pertain exclusively to the web browser and not the web template itself.

In addition, the '*Display for others*' and '*Duplicate*' low-level tasks were removed from '*Use Information*' because these are activities that are out of the scope of a web template. Namely these tasks entail the user showing someone else content from a web page in the former case, and copying content and pasting it on other applications in the latter.

Most of the original subcategories on '*Locate on page*' are irrelevant to web templates and were consequently removed from the model. For example, content that is "interesting" to the end-user of a web page or related to a specific concept are notions we cannot even predict and have no place in a template. The type of content that can indeed be found within a web page and also relates to web templates in particular are images, hyperlinks, thumbnails, video, audio, polls, animations, forms and fields. Most of these concepts are self-explanatory but distinctions are made between forms and fields because there are cases when a template may include a complete form integrated by a set of fields, labels

and buttons, for example a payment form; whereas other times there may be tasks that just require a field, for example executing a search function.

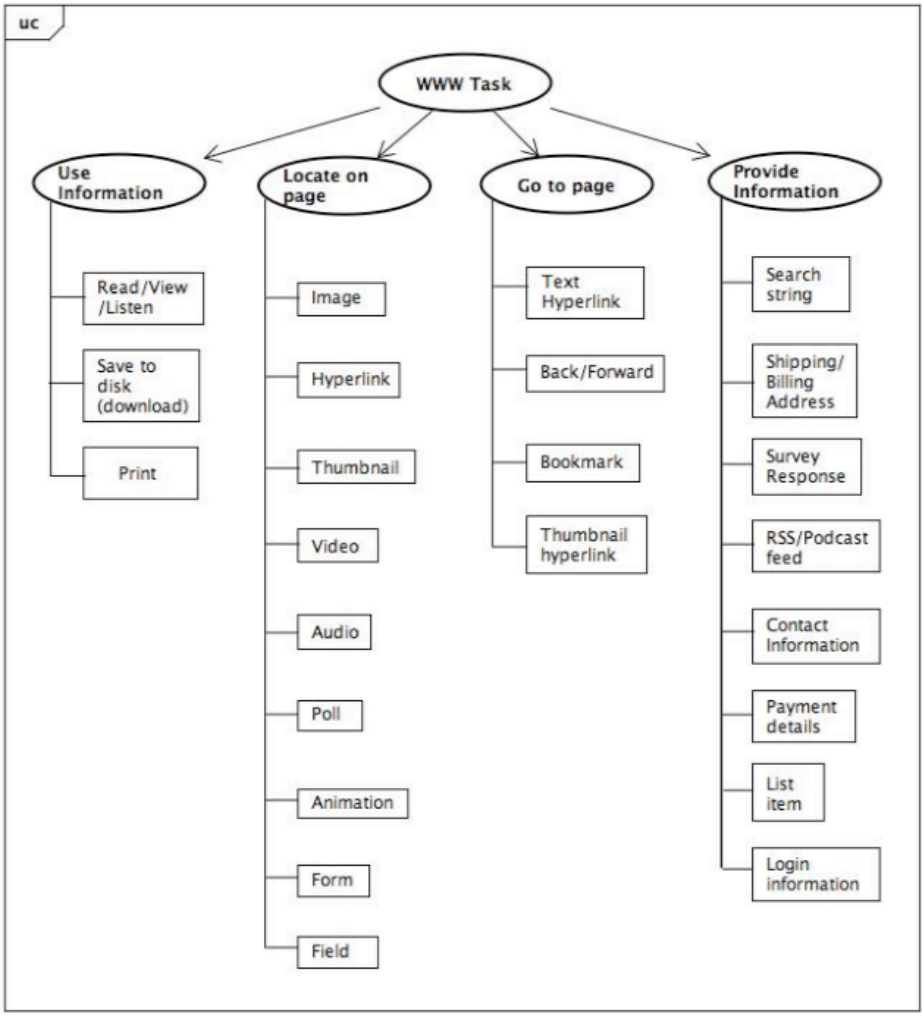

Figure 4.2: Task Model

The subcategories removed from '*Go to page*' were '*History list*' and '*Provide URL*' yet again because these tasks are the obligation of the browser (selecting an item from a list of previously visited pages and typing a URL into the address field of a browser). The 'Hyperlink' subcategory was split into two different concepts in order to specify whether the hyperlink comes in the form of text or of a thumbnail. The '*Back/Forward*' subcategory in this model does not refer to browser functionality, but to the navigational structure being part of the template, whether it is in the form of the words 'back' and

'forward' as hyperlinks or in the form of images clearly implying such functionality. A similar case applies to the '*bookmark*' subcategory where it refers to a section of the template designated for bookmarks as opposed to the bookmark list that is integrated to web browsers.

All the subcategories from '*Provide information*' were preserved with some changes and additions. First of all, the subcategory '*Shipping address*' was transformed into '*Shipping/Billing address*' in order to be able to apply it to either of these concepts. Moreover the subcategories '*RSS/Podcast feed*', '*Contact information*', '*Payment details*', '*List item*' and '*Login information*' were added, because they represent commonly performed activities in many web pages, for example those concerning e-commerce or entertainment.

4.1.1 User-Defined Tasks

The task model proposed offers a fixed range of the most common activities that end-users can accomplish on a web page template, however the description of these tasks can be further refined by the descriptions contributed by the users of the system, who in this case are software developers. This type of tasks is also included in the system, due to the fact that the vocabulary that a group of people may use to define a task can be very dissimilar from one another. Consequently, by having a task described in different ways, it will be easier to locate a web template that includes it during future searches of the repository, because it is very likely that different people will have differences in how they express the descriptions of a web template.

The structure by which the user-defined tasks are to be constructed is that of a verb and an object. This simple construction can accomplish an accurate and to the point description and, at the same time, equivalent tasks can be created by obtaining the synonyms for both the verb and the object; thus, providing a larger range of vocabulary to describe web template tasks. For instance, if the system allowed free-form descriptions of tasks, like in the survey presented in Section 2.2, the users would more likely enter sentences like "*The template provides a picture preview function*". A sentence like this provides unnecessary and redundant information: the assumption that the description is made in reference to the template is already explicit and is not required to be stated yet again. In the verb and object form, this sentence could be expressed simply as 'verb: *preview*, object: *picture*' and it still conveys the same idea in a structured form.

4.1.2 Synonyms

With the goal of further enhancing the task descriptions, the web templates should also be annotated with synonyms of the tasks defined by the user, in order to have descriptions that contain additional potential search terms that users could employ when

making a search. These synonym tasks follow the same verb and object structure as the user-defined tasks so that they can be retrieved from a lexical database and still convey the same idea as the original user-defined task. Moreover, as pointed out in Section 2.3, the WordNet service is used as the source from which these synonyms are retrieved.

WordNet, in many cases, provides synonyms for more than one sense or meaning, so for this reason not all of the words supplied by the service will necessarily apply to the task being described. As Figure 4.3 demonstrates, a search for the word '*browse*' returns seven sets of synonym results, namely three for the word as a noun, one where the meaning of the word is vegetation, another where it means to read superficially and a third about the act of feeding by nibbling; and four as a verb, one that refers to shopping, another about feeding in a meadow, a third about looking around and the fourth in regards to eating lightly. Each of the results represents a different meaning, or sense, of the word, which is explained within the glossary that is displayed next to the synonym set. This glossary helps users avoid confusion as to what synonym set is the appropriate one for the idea they want to convey.

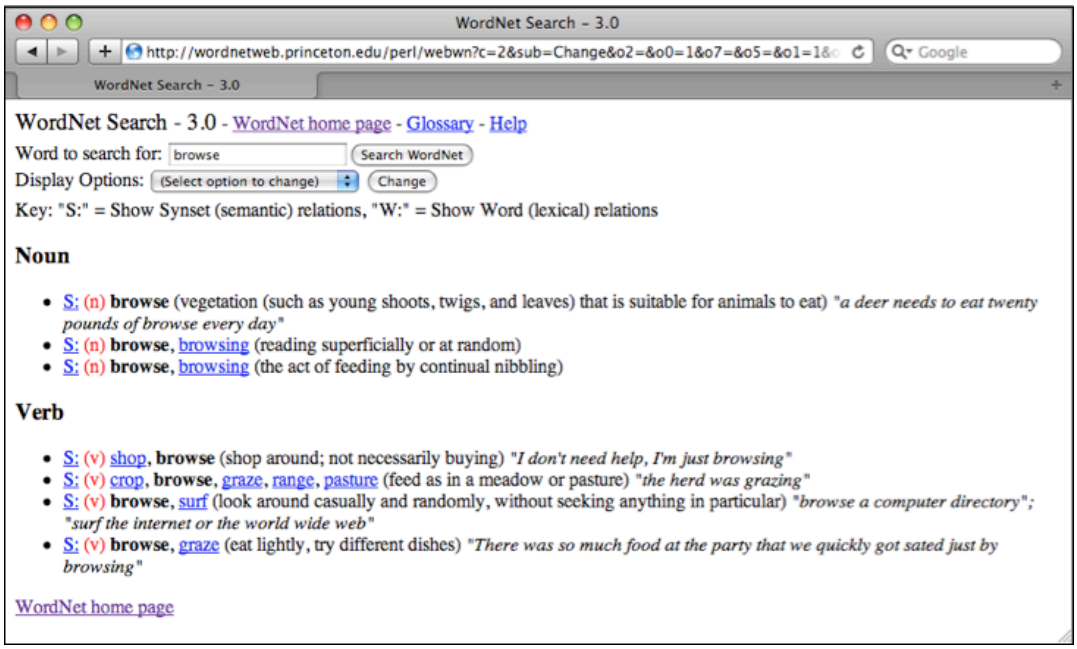

Figure 4.3: WordNet search results for the word '*browse*'.

Within our approach, the user of the system does the disambiguation of the sense of the word and he or she shall select the synonym that more precisely fits the task description being expressed. For instance, when describing an e-commerce web template, a user is very likely to choose the synonym '*shop*'; whereas the selection of the synonym '*surf*'

might be better suited when the description deals with a template that offers a set of hyperlinks to browse into other web pages.

4.2 Additional Annotations

The remaining elements of the template model provide a higher level of detail in the representation of a web template in an attempt to emphasize the features that distinguish them from one another.

4.2.1 Theme, Context and Device

According to [Ken00], the theme of a web site should be presented visually and not just through its text contents, therefore the subject matter of the content of a web page normally influences its graphic design. For instance, the example of a web page that offers games for children seen in Figure 4.4 uses a variety of images with bright colors and cartoon-like characters, which appeal to its target demographic.

Figure 4.4: Children themed web template.

On the other hand, a web page for an educational institution (Figure 4.5) displays a different visual approach in order to convey professionalism and seriousness, because the target audience of the web template are people looking to pursue a higher education

degree and are most likely to be engaged by an institution that conveys the message of taking their education seriously.

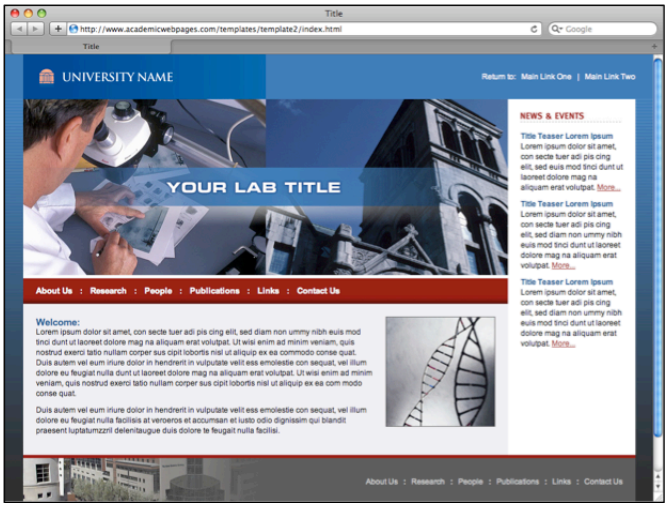

Figure 4.5: Education themed web template.

There are times however, when a web template can be defined using more than just one theme as in the example shown in Figure 4.6: a web page about education for children. In this case, with the combination of two themes, the web template is aimed at two very different groups, that is, children and their parents. The web template uses a style that is professional enough to appeal to the adult population, but that still provides colorful illustrations to appeal to their younger target group.

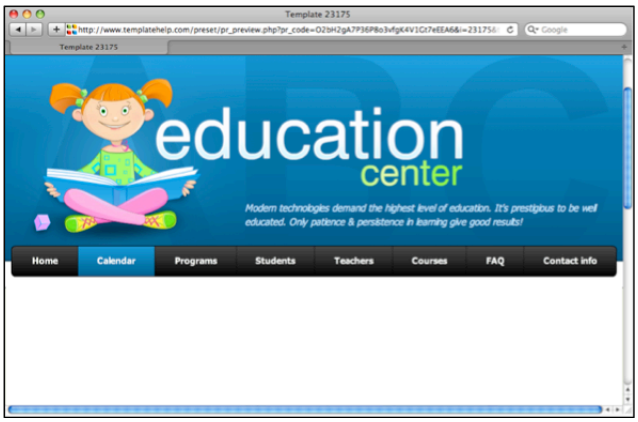

Figure 4.6: Web page template with more than one theme.

After reviewing these examples, the best solution is to allow the annotation of the theme of a template to be of at least one but without a top boundary. This way we can allow for template descriptions to be more definitive taking into consideration that one web template may be designed for more than one theme.

Another characteristic that can differentiate a web template, in spite of having the same theme others, is the context under which it is used. For example two web templates about books can accomplish different tasks when one is used under a personal context, for instance a blog about books like the one seen in Figure 4.7, and the one in Figure 4.8 as an e-commerce application.

Figure 4.7: Web template of a blog about books.

The bookstore web template (Figure 4.8) allows its end-users to add items to a cart, browse different literature genres and create a shopping account among others. On the other hand a book blog lets its visitors read text, view images and sign in. Thus the tasks each of these common-themed templates is able to perform are quite different between one another.

The contexts defined for use in the annotations are:

- *E-commerce.* This context describes web templates where commercial transactions can take place. Such as online stores and banking applications.
- *Business.* This concept applies to templates that publish information about a corporation or enterprise but where transactions do not take place.

- *Personal*. This is the kind of web page authored by individuals.

Figure 4.8: Bookstore web template.

So far the examples provided show web templates meant to be rendered on computer web browsers, but the ubiquitous computing model [Wei93] encompasses the use of different devices that differ in size and connection speed. Therefore, accessing the Internet through these different devices will also require different types of web templates. Some examples of presently used ubiquitous computing devices are mobile phones and smart phones and, in order to access web pages through them, the web pages need to be adapted to fit to smaller screens and to limit their content to adjust to limited connection speeds. Hence the device attribute should also be included in the annotations of a web template.

4.2.2 Structure, Widget/Elements and Position

Widgets and other visual controls such as labels, text boxes, text areas, checkboxes, lists and push buttons among others, play an important role into the functionality that can be achieved through a web template. For example, when a developer creates a method to sign in to an application, the Graphical User Interface will require not only the space and ability for text and images to be displayed, but typically it will also involve two text fields and one push button.

The position of the graphical controls within the web template is also a useful piece of information because it could be a sign of whether the focus of the template is the

functionality provided by these widgets or if it is just a small part of it, like when they are placed within a frame (Figure 4.9). The location of the fields also hints to the functionality, for example, the '*username*' field usually comes before the '*password*' field and not the other way around.

Figure 4.9: Web template with user login functionality.

5 Web Template Retrieval

As discussed in the previous chapter, the annotations that are added to a web template, serve the purpose of giving it a detailed description so as to be able to distinguish it from others. In order to enable the retrieval of a web template that will meet specific needs according to the requirements of a given application, a search mechanism that can take advantage of these descriptions is essential. A search functionality represents a time-effective solution because, in this case, the users of the system will not need to browse through the repository and analyze the description of each template in order to find the appropriate one. Instead, by submitting the characteristics of the web template they need, the system will query the annotation repository and return results that match their demands.

The overall architecture of the task descriptions of a web template is presented in Figure 5.1. A web template can perform a series of tasks, which can also be expressed by using synonyms.

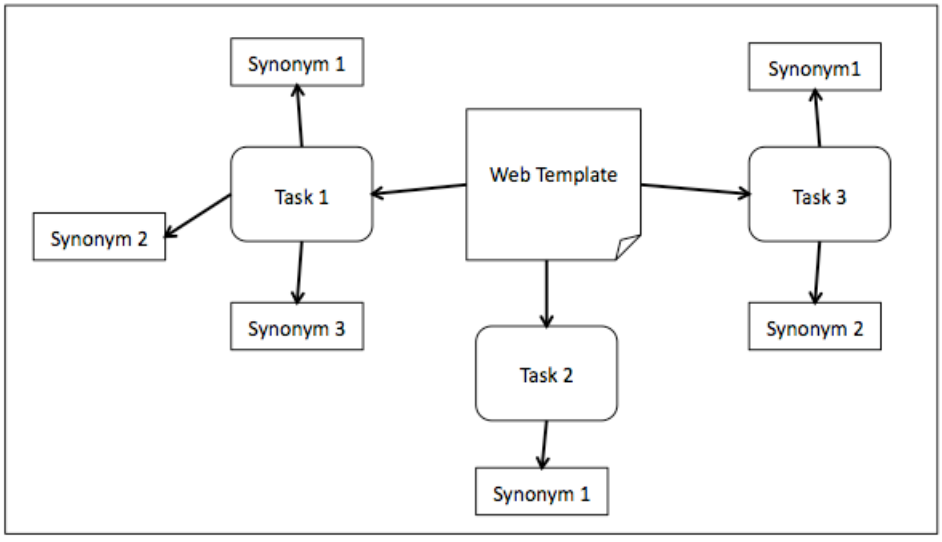

Figure 5.1: Web template, task and synonym relationships.

This chapter presents an overview of the specific characteristics to make the web template

retrieval an effective process in Section 5.1, while Section 5.2 offers a detailed explanation of the search algorithm to accomplish this.

5.1 Requirements

This section will explain the parameters that need to be considered by the search algorithm along with the roles that the task annotations and synonyms in particular ought to play in it, to be able to provide results in accordance to user needs.

5.1.1 Search Parameters

First of all, in order to take full advantage of the information provided by the annotations of the web templates, the search mechanism should give users the opportunity to query the repository based on all of these metadata. Therefore, the search interface, rather than provide only one search parameter to query the repository, should instead make available to the user several parameters for each of the annotation properties. This way the repository can be queried based on one, all or whichever combination of these parameters, the user chooses according to his or her specific needs. The parameters that can be used to query the template repository are:

- Title
- Tasks
- User-defined tasks
- Device
- Theme
- Context
- Layout

It is important for the interface to also allow the user to enter more than one task from the task model presented in Section 4.1, as well as multiple user-defined tasks. This way, the algorithm is given the capability of searching for templates that are able to perform all of the desired functionalities searched by the user.

5.1.2 Task and Synonym Relevance

The tasks that a web template is able to perform play a very important role both in terms of annotation and retrieval, because these tasks are directly related to the functionality

of the application for which a developer seeks a template. Describing the functionality that can be achieved by a given web template allows for it to be identified for its reuse in future applications that need to execute the same tasks.

However, given the fact that several users will submit, annotate and retrieve web templates, the vocabulary they will use to describe a task is very likely to vary. For this reason, the use of synonyms is proposed to allow for alternate and equivalent ways to describe a task with the purpose of making them discoverable by people who use different vocabulary. For example, in order to describe a web template that allows the reproduction of a video file, different ways to describe this function could be made, like 'verb: *run*, object: *movie*', 'verb: *play*, object: *picture*' or 'verb: *show*, object: *video*'; all of which, convey the same idea, so they are considered equivalent descriptions. So in order to cover the differences in vocabulary, the annotation process should be a collaborative effort.

As part of this collaboration, it is important to allow users to express their agreement to previously assigned user-defined task descriptions and synonyms as a way to represent how closely a task or synonym is associated to a web template. For instance, two different web templates may be annotated with the task 'verb: *play*, object: *slideshow*'. One of these templates, *template A*, is integrated by a series of thumbnails and the other, *template B*, by a flash animation that loops through a series of pictures. After analyzing both templates a group of five people could agree that the latter template is correctly described by the 'verb: *play*, object: *slideshow*' task, however, they don't agree that it necessarily applies to the former. After considering these opinions, we can say that *template B* is a better match for the task of playing a slideshow and, if a user were to search for a template that can carry out this function, then *template B* should be ranked higher than *template A*.

The evaluation of the correctness of the associations between tasks or synonyms and templates can be accomplished in the form of a voting function where, out of the tasks and synonyms that have already been assigned to a template, a user can select the ones that they believe more accurately describe the functionality provided in order to give these selections a higher relevance over the rest. Every time a user selects a user-defined task or a synonym, the selection made will earn one vote.

The voting mechanism can also allow us to recognize which description is more appropriate for a template within a user community and this allows for the possibility of having a synonym task to be voted more often than a user-defined task. In this case, the synonym should take precedence over the task because the template is more likely to be searched for using these terms rather than the originally described task, and a reversal of their roles should occur. Hence, the synonym becomes the task and vice versa. For example, as illustrated by Figure 5.2, first a web template is annotated with a task (T), and a selection of synonyms (S1, S2 and S3) is made out of the possible alternatives returned by WordNet. Initially both the task and the synonyms are all assigned one vote.

Subsequently, when another user encounters this template, he or she selects synonym S1 as the better description for the task, thus increasing its vote count by one. After this voting takes place, the system evaluates the vote count for task and synonyms and, upon finding that the count for S1 is higher than for T, the roles between these two descriptions are reversed, thus S1 becomes a task and T turns into its synonym.

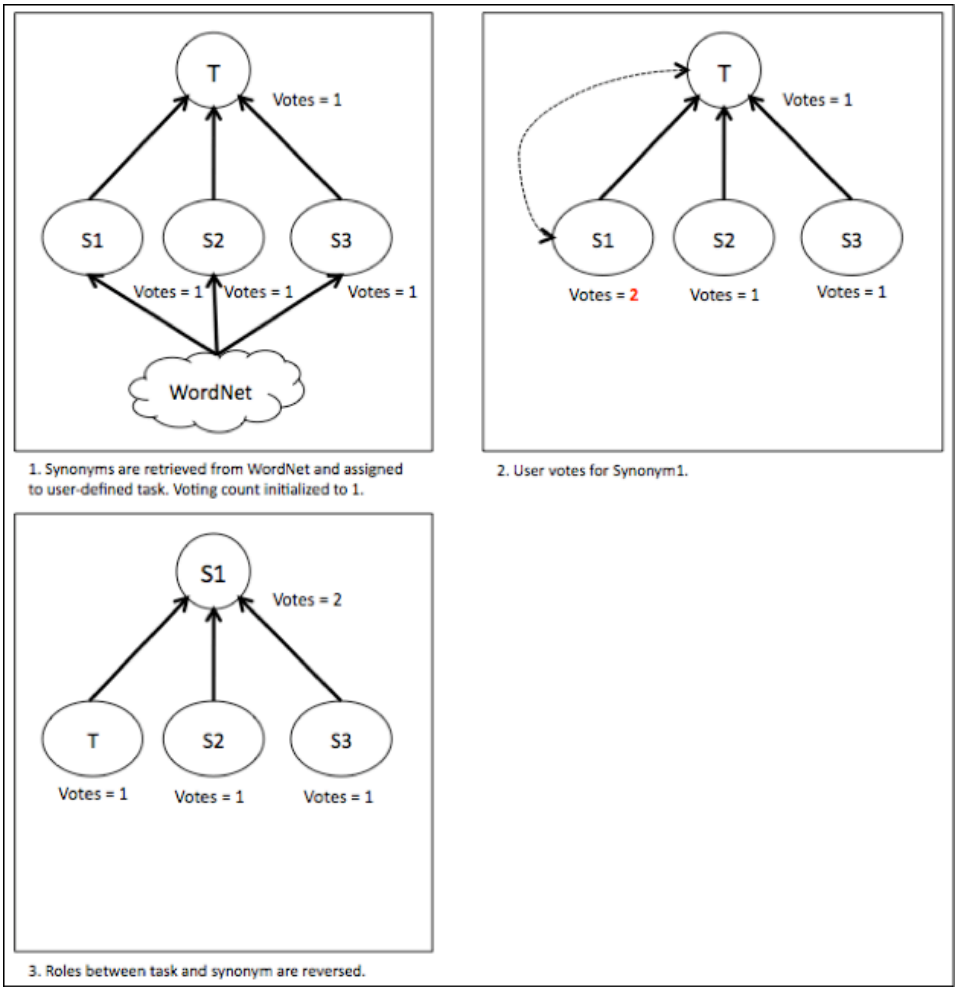

Figure 5.2: Task and synonym role reversal.

5.1.3 Result Ranking

There is a likelihood that a search may yield a large number of results that match the parameters provided. In order to potentially save the user time, a ranking mechanism

should be part of the search algorithm because, even though several web templates may be described with the same task, some will have more support from the user community in terms of the accuracy of the description than others. As described in the previous section, users should be allowed to evaluate the correctness of the user-defined tasks and synonyms by voting on the better-suited options and this vote count should be incorporated into the computation of the score of the web template during search, in order to return the results that better match the search criteria.

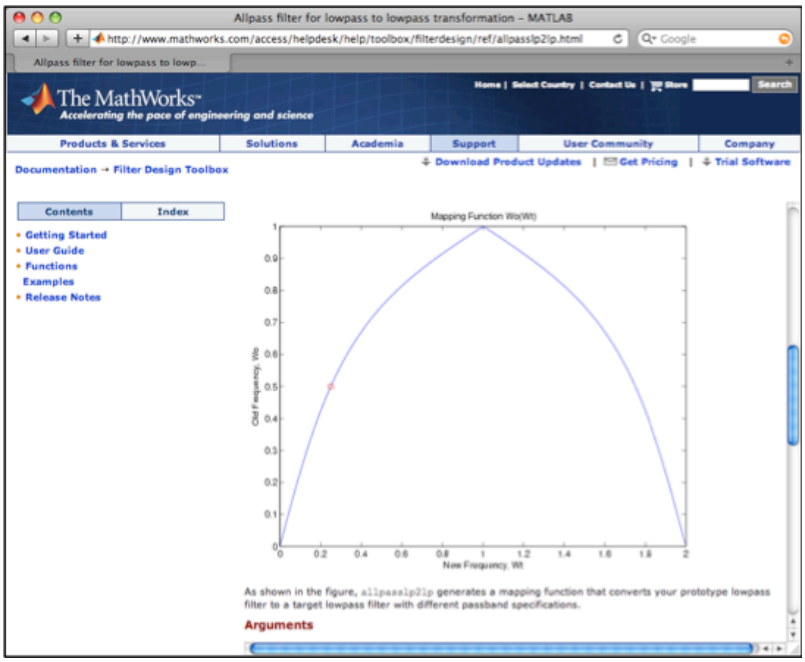

Figure 5.3: Web page displaying the graph of a mapping function.

For example, a user needs a web template that can be rendered in a computer browser to be used in the context of business and that provides the capability of displaying a street map and performs a search in the repository. The criteria are entered as 'device: *computer*, context: *business*, verb1: *display*, object1: *map*, verb2: *read*, object2: *text*'. In the repository there are two web templates that successfully match the given parameters: the first web template (*WT1*) was designed for use in computer browsers in a business context and is able to display text and graphs for mathematical mapping functions (Figure 5.3), for example a graph for a function that is applied to a set of values; the other template (*WT2*) shares the same device and context characteristics as *WT1* but this one is capable of showing a street map (Figure 5.4) instead of the graphical representation of a mathematical function. Both of these templates meet all the requirements according to the criteria submitted and could potentially have the same score even though, given

the user needs, the better result is *WT2*. With the voting mechanism in place, it is very likely that several users will vote on the 'verb: *display*, object: *map*' task for *WT2* but not for *WT1*, if the word "*map*" in this user community is more frequently used to refer to street maps instead of formulas. Consequently, by factoring the number of these votes into the ranking mechanism, *WT2* will be presented to the user as the better result option for his or her query.

Figure 5.4: Web template displaying a street map.

Thus, to calculate the score of a web template result, one point should be granted for each parameter that it successfully matches to the query data. For example, if a user queries for a template with the user-defined task of 'verb: *display*, object: *map*', the templates described with that user-defined task will earn one point in the user-defined task area. Additionally, the user-defined task points should be multiplied by a factor to give them a weight of 50% higher than the rest of the annotation properties in order to make these results more representative because the functionality that a web template is able to perform is the most important aspect in order to use the templates that match the functionality of the application that the users develop. Furthermore, the number of votes, which represent the number of times a user-defined task has been selected by a user as a good way to describe a function the web template can perform, is multiplied by

the user-defined task points, so as to consider the user voting, which signifies the level with which the template fulfills the task. Finally, all these values are added. Therefore, the formula is constructed in the following way:

$$Score_{WT} = A_1 + \ldots + A_N + \sum_{1..N} T + \sum_{1..N} ((UDT * 1.5) * F)$$

In this formula $Score_{WT}$ represents the total score of a web template, while $A1...N$ correspond to the annotation properties for *Title, Device, Context* and *Theme*. The $\sum T$ denotes the addition of values for all tasks searched from the task model and UDT stands for all the User-defined tasks searched. Finally, F represents the frequency of a User-defined task.

Taking the example of the template that needs to display a street map and considering that the only criteria that the user entered for the search were for the annotation properties of device, context and two user-defined tasks, we have the following resulting formula:

$$Score_{WT} = A_1 + A_2 + \sum_{1..2} ((UDT * 1.5) * F)$$

Considering that *WT1* meets the criteria for the annotation properties of device and context and that the user-defined task of 'verb: *display*, object: *map*' was voted as a good description for the task only once, hence giving it a voting frequency of 1; while the task of 'verb: *read*, object: *text*' has a voting frequency of 3 we can fill in the values for the formula as follows:

$$Score_{WT1} = 1 + 1 + (((1 * 1.5) * 1) + ((1 * 1.5) * 3)) = 8$$

Meanwhile *WT2* also meets the criteria for the annotation properties of device and context but both the user-defined tasks of 'verb: *display*, object: *map*' and 'verb: *read*, object: *text*' have a voting frequency of 3, because three different people found this web template and had the opinion that these were good ways to describe the tasks it can perform. Hence the following:

$$Score_{WT2} = 1 + 1 + (((1 * 1.5) * 3) + ((1 * 1.5) * 3)) = 11$$

Thus, *WT2* gets a higher score than *WT1* and proves to be the better result for the query that the user entered.

Finally, the user should also have the possibility of stating the maximum number of results desired, which should prove helpful whenever the search finds a large number of matches.

5.1.4 Alternate Result Suggestions

With the intention of anticipating query scenarios where no results that match all of the parameters entered by the user can be found in the repository, the system should display a list of suggested results by taking advantage of the user-defined task synonyms. In this case, the originally queried verb and object should be looked up in the synonym database of the system, and a new query should be performed, in which the synonyms found replace the user-task description originally supplied by the user. This way, it is still possible for the users to obtain a set of effective web templates with the functionality they sought after but that were described using a different vocabulary.

In case that still no results are found, then the system should ultimately attempt to find web templates that meet any, instead of all, of the search criteria provided by the user. For instance, if a search for the parameters 'theme: *sports*, device: *computer*, verb: *display*, object: *picture*' does not yield any results at all, then a search with these parameters but using an '*OR*' Boolean operator should be executed. It is likely to, for example, find a template that also provides the ability to display pictures but for a different theme, in which case, only the theme portion of the content of the template would be changed but the developer would still not need to create a brand new web template for the application.

Consequently, by offering alternative results, the opportunity of web template reuse is still made possible by modifying templates that are as close as possible to the requirements of the application. Furthermore, the modified template can be later on added to the repository and annotated accordingly.

5.2 Search Algorithm

Considering the approach presented in the previous section regarding parameters, task and synonym relevance, result ranking and the suggestion of alternative results, the search algorithm should conform to the following sequence, once the user has entered the desired criteria.

1. **Query construction.** Upon receiving the search criteria from the user, the system constructs a query that uses a Boolean AND operator in order to retrieve web templates that meet all of the criteria.
2. **Search execution.** The system executes the query constructed in the previous step and calculates a score to rank the results.
3. **Check for results.** The system checks whether the query returned results. If it did, then step 9 takes place. In case there were no results found, step 4 is executed.

4. **Synonym lookup.** The system now looks up and retrieves the synonyms for the user-defined task that the user originally entered.
5. **Re-construct query.** The query is now re-formulated with the synonyms found in the previous step.
6. **Execute search.** The system executes the query and computes a score to rank the results.
7. **Check for results.** The system checks whether the re-formulated query returned results. If it did, then step 9 takes place. Otherwise, step 8 is executed.
8. **Alternate search.** The system now executes a query with a Boolean OR operator to find web templates that meet any of the criteria provided by the user.
9. **Display results.** The system displays the ranked results.

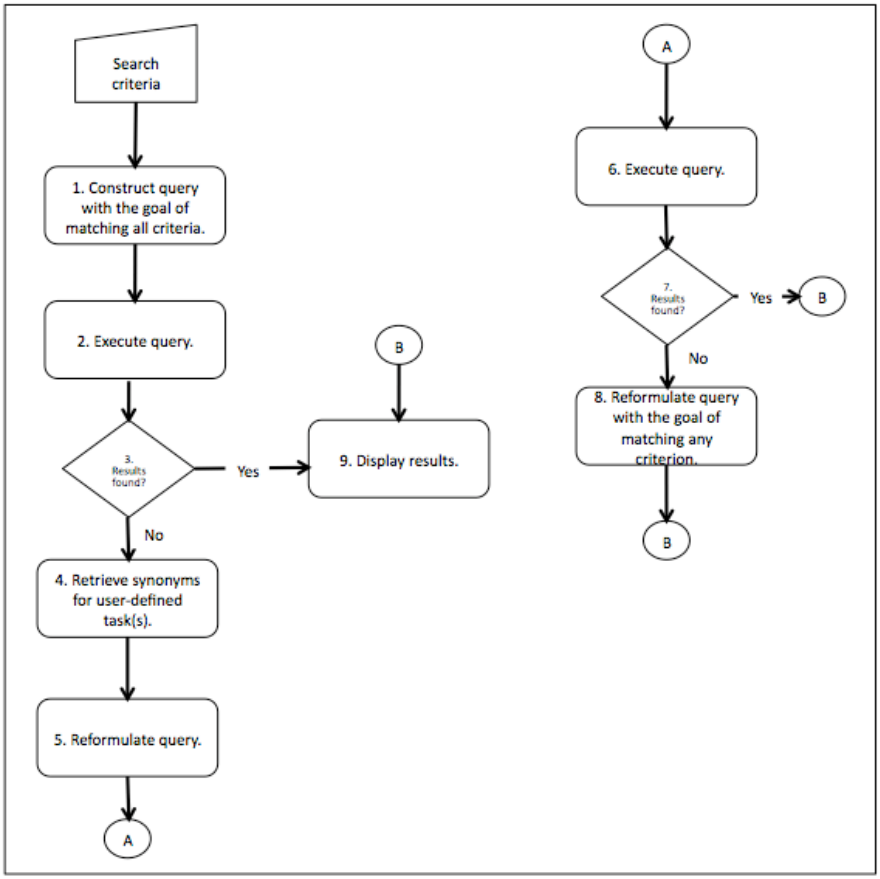

Figure 5.5: Search algorithm flow chart.

6 Implementation

The concepts presented in Chapters 4 and 5 are put into effect in the form a web-based application that enables the use of a repository that stores web templates along with its annotations. The application also provides the functionality of searching for the web templates.

The particular details regarding the implementation of the system are discussed in this chapter and are organized in the following way. First, Section 6.1 explains the *Template Model* implementation aspects, while Section 6.2 presents the details regarding the *Annotations* that describe the web templates. Subsequently, Section 6.3 offers an explanation of the *Web Template Scoring* functionality, followed by Section 6.4, which gives a detailed explanation of the *Search Mechanism* used to find web templates in the repository. Finally, Section 6.5 provides a set of examples of the *Operations* that can be executed in the system.

6.1 Template Model

The template model introduced in Chapter 4 is at the core of the system because it represents the structure of the annotations of a web template. The implementation of the model includes all the relevant information to a web template so as to be able to accurately describe it and allow for its future retrieval.

As presented in Figure 6.1, the '*LayoutTemplate*' entity lies at the center of the structure and, besides the data to allow it to be connected to other entities, it includes a '*name*' attribute to allow the template to be identified easily by the user as opposed to using only a numerical ID. Additionally, the '*title*' attribute is used to denote the role that the template plays within a website. For example, a template could specifically have been designed to be used as an index page, a blog or the '*about*' section of a website. A template can be associated with several themes, tasks and user-defined tasks; however, it is limited to one context, one structure, one device and one screenshot. With the purpose of enabling a graphical preview of a web template in the system so that the users can be sure that the template is the correct one before they download it, the '*Screenshot*' entity is included as part of one the implementation features. This entity is separated from '*LayoutTemplate*' to ensure that the database table in which this information will

be stored is normalized [RG02] because the '*Screenshot*' entity holds additional data that is dependent on its own key rather than on the key of the '*LayoutTemplate*' entity.

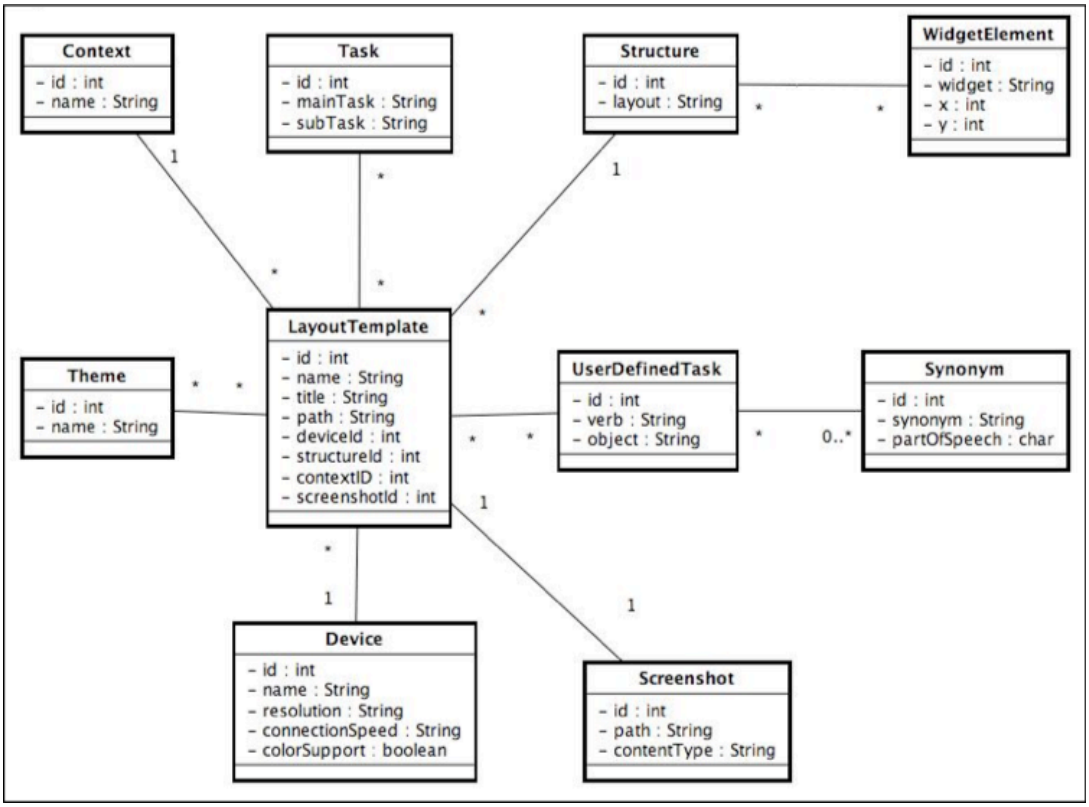

Figure 6.1: Class diagram representing web templates.

In addition to the name that identifies a device and in order to further specify its technical characteristics, the '*Device*' entity also allows the storage of data in regards to its resolution, its connection speed and whether or not the display of the device supports colors. This entity and the details it provides will allow the user to know whether or not a template can be used on the targeted device for the application he or she is developing because, as explained in Section 4.2, presently, web applications are developed not only for computer web browsers but for different types of mobile devices as well.

The entity '*UserDefinedTask*' is integrated by a verb and an object, as defined in Subsection 4.1.1. The '*Synonym*' entity does not have the same structure as '*UserDefinedTask*' even though it is used to represent a synonym task. Instead, the '*Synonym*' entity holds the attributes of synonym and part of speech. The synonym attribute is a single word that can be either a verb or an object and the part of speech attribute stores which one of those roles the word plays, which can be either a verb or an object.

The reason to implement the synonyms with this structure is to have the ability to handle two possible scenarios. In the first, synonyms are associated to either just the verb or the object but not to both. For example, the synonyms that WordNet currently returns for the word '*navigate*' are '*voyage*', '*sail*' and '*pilot*', which do not have the meaning for which the original word would be used in a web template, which could be '*browsing*' or '*surfing*'. Thus, no synonyms would be selected for this verb but it is still possible to make selections for the object. The second scenario contemplates the possibility that the users may select a different number of synonyms for a verb than for an object. For example, for the task 'verb: *listen*, object: *audio*' (Figure 6.2), it is possible to select one synonym for the verb, like '*hear*', and two synonyms for the object, for instance '*sound*' and '*recording*'. Therefore, the synonym tasks can be any combination of these words, that is, 'verb: *hear,* object: *sound*' and 'verb: *hear*, object: *recording*'.

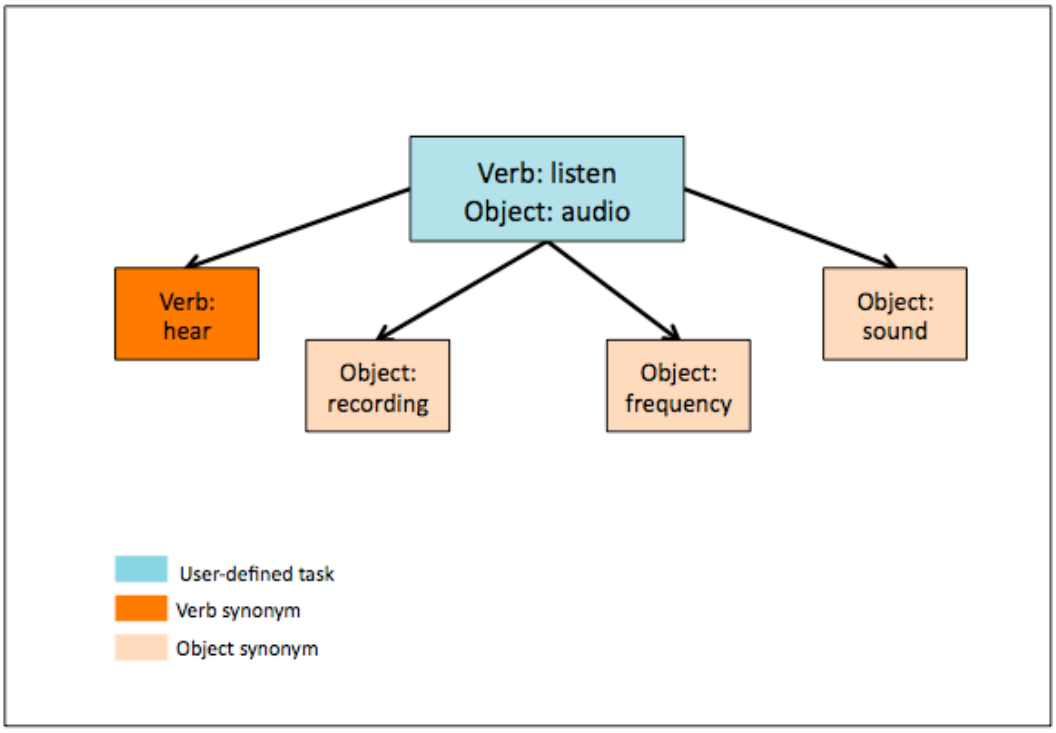

Figure 6.2: User-defined task surrounded by its synonyms.

The annotations denoted by the model in the previous section are stored in a *MySQL* database with the structure shown in the Entity relationship diagram in Figure 6.3. This diagram presents all the entities in squares and the relationships that exist between them in rhombi. The relationship '*annotated with*' does not just include the identifiers of the '*Layout_template*' and '*User_defined_task*' but the '*Frequency*' property as well to hold

the votes that a user gives for a user-defined task as explained in Subsection 5.1.2. The relationship '*has*' also contains a '*Frequency*' attribute, but this one holds the number of votes that users give to synonyms.

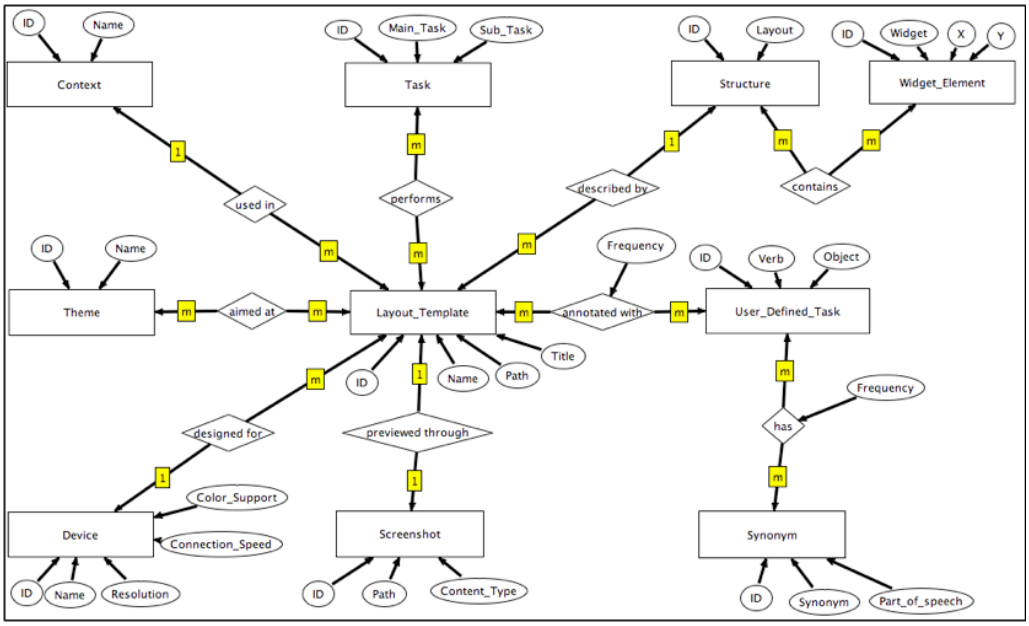

Figure 6.3: Entity-relationship diagram.

6.2 Web Template Annotation

The annotation of a web template takes place at the same time as it is submitted to the repository to ensure that it will have a description through which it can later on be searched for. In addition, the users can edit the annotations for templates stored in the repository.

The interface used to submit and annotate new web templates, is presented in Figure 6.4 and it allows the user upload a web template file and a screenshot to the repository and also to enter the description of the template as: ID, title, device, theme, context, structure, task and user-defined task. In the particular case of theme, task and user-defined task, multiple entries are permitted for each criterion because a web template may be able to perform more than one task.

A similar interface is provided to allow the editing of existing web templates. The interface for edition contains the same parameters, except for the ability to upload a

screenshot and a template file. In addition, when the editing interface is started, it displays the existent annotations for the selected template.

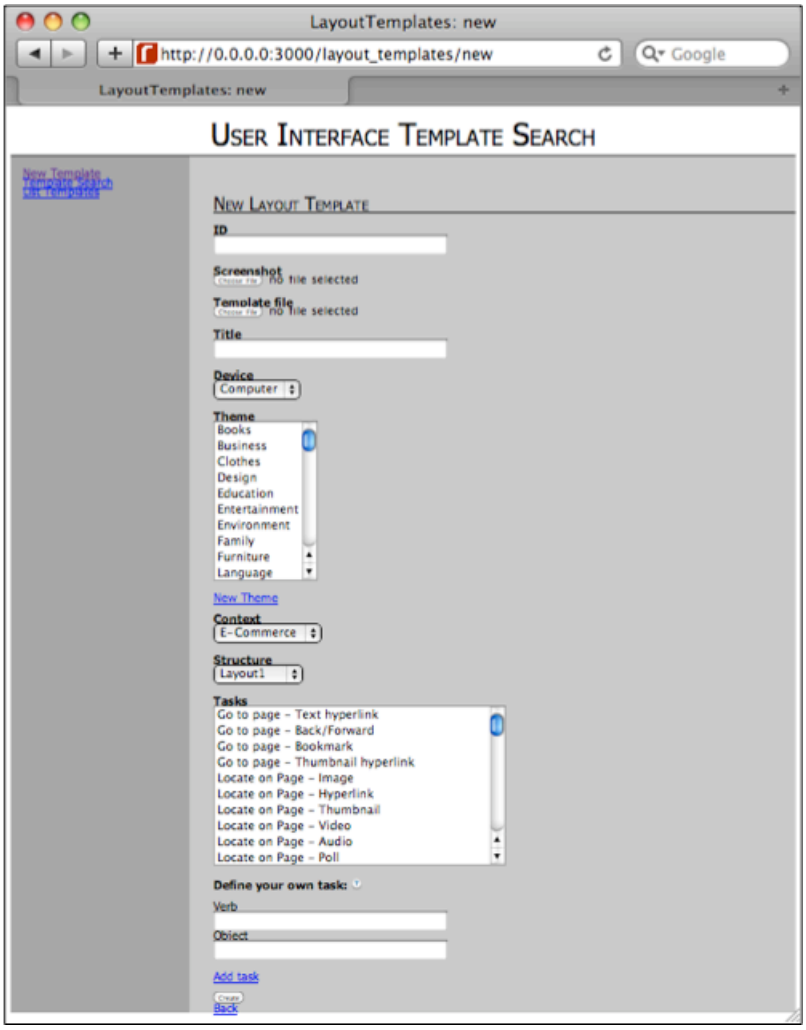

Figure 6.4: New web template interface.

6.2.1 User-Defined Tasks and Synonyms

In the specific case of the user-defined task annotations, the interface provides the possibility of stating the annotation as one verb and one object; and when the user decides to add the task to the template, the system looks up the words entered in the WordNet ontology. Finally, the system displays a list of the synonyms it finds for each

of the words along with the glossary information for them. Consequently, the user can choose all the options that apply to their task and save those selections (Figure 6.5).

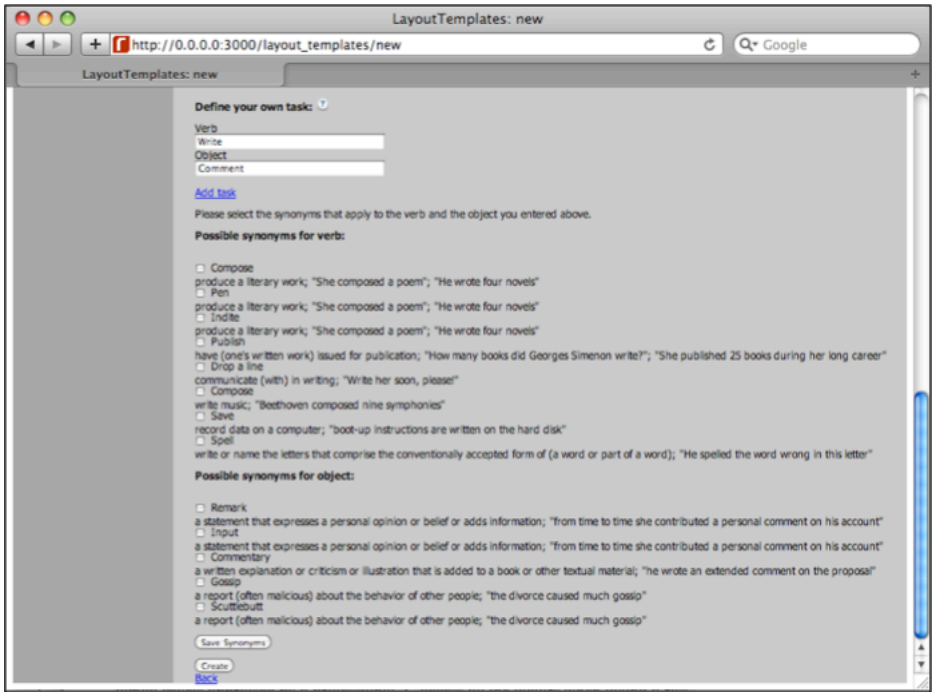

Figure 6.5: New web template interface after synonym lookup.

User-defined tasks, which are integrated by two words, and synonyms, which entail only one word, need to have the number of votes they received accumulated in order to help determine their relevance to the web template they are assigned to, as well as to aid in search result ranking calculations. To this effect, each word from a user-defined task or a synonym will have a value of one point upon creation and will be incremented by one whenever a user gives a vote to it. Given that user-defined tasks are composed by two words and each word has a value of one point, their initial value will be of two. Likewise, whenever a user votes for a user-defined task, the frequency value increases by two points. This is done in order to establish a fair calculation of the relevance given between user-defined tasks and synonyms, which was presented in Subsection 5.1.2, because the relevance score of a synonym task will be calculated by adding the frequency value of the verb of the synonym task and the frequency value of the object of the synonym task. For example, Figure 6.6 presents the initial value of a user-defined task, 'verb: *listen*, object: *audio*', which is 2 because it is integrated by two words. In comparison, the synonyms that are associated to it, hear, recording, frequency and sound, each has an initial value of 1, but once the synonyms are combined to form tasks, '*hear recording*',

'*hear frequency*' and '*hear sound*', the value of each word is added and the each synonym task also has the value of 2. Hence, the user-defined task and the synonym tasks have the same relevance in this case.

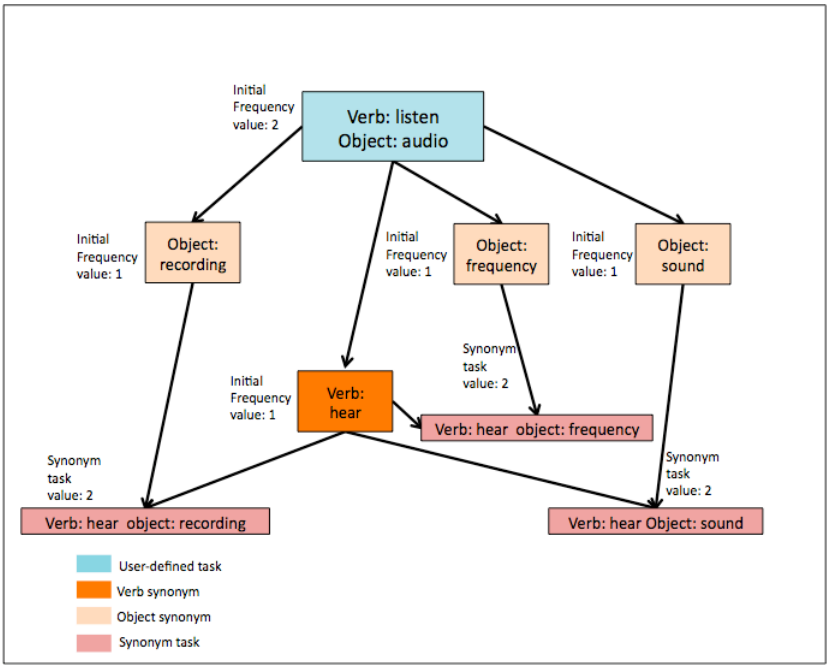

Figure 6.6: Initial frequency values of user-defined tasks and synonyms.

6.2.2 WordNet Integration

The process of obtaining the synonyms for the user-defined tasks is made possible by the use of the lexical database WordNet and, as explained in Section 2.3, it can be accessed via an online service. In order to establish a local interface into WordNet, however, several APIs have been created, for instance for Java, Ruby, .NET, PHP and Perl, which can be used in conjunction with different technologies. In order to integrate this lexical database into the system, the Ruby-WordNet API was used.

The technical requirements for the Ruby-WordNet API are the WordNet 3.0 data files and a conversion of these into a Berkeley[1] database, which is an open-source key-value database that comes in the form of a library and can be embedded to an application in order to make function calls to retrieve data instead of contacting a remote server. The conversion process is accomplished by running a script provided by the Ruby-Wordnet

[1] http://www.oracle.com/database/berkeley-db/index.html

page. Finally, to access the database, Ruby-WordNet is installed using a ruby gem, which is an installation package that only needs to be executed to perform an install of a library without the need of additional input from the user.

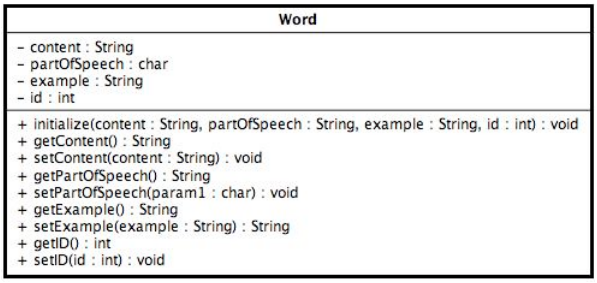

Figure 6.7: Word class.

In order to retrieve the synonyms for a given word, the system accesses the WordNet lexicon by creating an instance of the '*Lexicon*' class and uses one of its functions to retrieve all the available synonym sets along with the glossary information. The API also offers a synonym set class, called Synset, which provides a function that returns all the words that integrate a synonym set. In order to have more control of the synonym data between the process of retrieving it and the process of displaying it on the system, the '*Word*' class was created (Figure 6.7) in our system.

The '*Word*' class stores a synonym, the fact of whether it is a verb or an object, its glossary and a numerical ID.

The overall process of obtaining synonyms is the following. First, the system creates an instance of the WordNet Lexicon class and supplies it with the words that the user entered, that is, a verb and an object. The API returns a group of synonym sets, one for each meaning of the word supplied, and the system traverses all of the sets to extract each of the words in them separately. Consequently, the system creates an instance of the '*Word*' class and stores the synonym, its glossary and the character '*v*' or '*o*', depending on whether the synonym is a verb or an object; additionally, the system assigns the synonym a numerical ID. Finally, the instance of '*Word*' is saved to an array, which is then passed to the interface to display the results to the user.

6.3 Web Template Scoring

An interface (Figure 6.8) is provided by the system to allow the users to download a web template, which in addition, lists all of its annotations and displays a screenshot for the file.

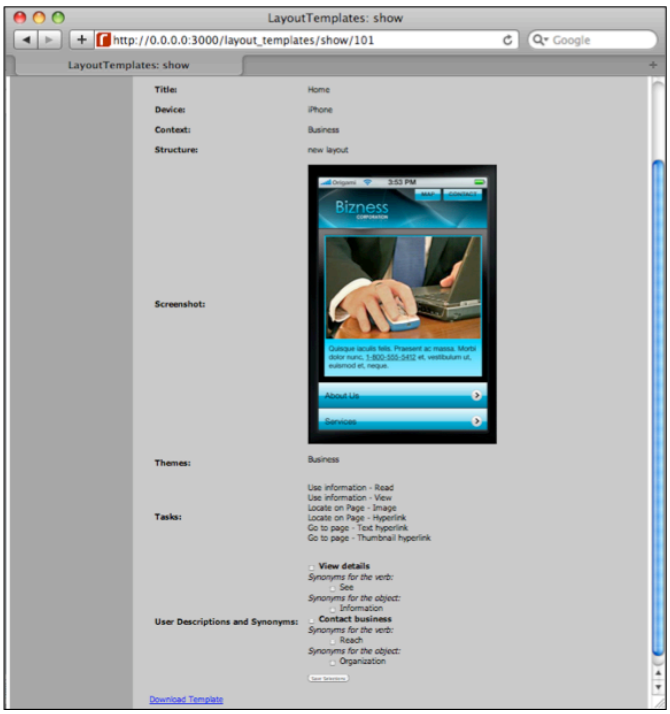

Figure 6.8: Web template download page.

This interface also allows the users to vote regarding the existing user-defined tasks and synonyms for the template. Each user-defined task along with its synonyms is displayed next to a checkbox for the user to select the items that, in his or her opinion, presents an accurate description of the tasks that the template can accomplish.

Once the selections have been submitted, the system will update the frequency of the polled items and will perform a check to find out whether the frequency of a synonym has reached a higher value than that of a user-defined task. The check consists in creating synonym tasks out of all the synonyms assigned to a user-defined task, by creating all possible combinations of verb/object pairs, and comparing their joint frequency value against the frequency value of the user-defined task. First of all the system retrieves the synonyms and stores them in two arrays, one for the verbs and another for the objects. Then, as illustrated in the pseudocode in Listing 6.1, both arrays are traversed to create all possible verb/object pairs and the system adds the frequency of the verb plus the frequency of the object. Subsequently, the system compares the result against the frequency of the user-defined task and, if the added frequency of the synonyms is greater than the frequency of the user-defined task, then the roles of the synonym task and the user-defined task are switched.

```
0 Method check_relevance(UserDefinedTask userTask, Synonym verbs[], Synonym
      objects[])
1    for each verb in verbs[] do
2       for each object in objects[] do
3          synonymFrequency = verb.frequency + object.frequency
4          if synonymFrequency > userTask.frequency then
5             switch_roles()
6          end if
7       end for
8    end for
9 end
```

Listing 6.1: Relevance check method pseudocode.

Figure 6.9 presents an example in which, after an initial state where the user-defined task and the synonym tasks had equal values, two of the synonyms have received one additional vote each that was given by a user who deemed the synonyms as a more representative way of describing the task that the web template performs. The synonym words are the verb '*hear*' and the object '*sound*' and after they each receive one vote, the value of the synonym tasks created by combining both of these words reaches a value of four. Meanwhile, the user-defined task received no votes, which means that its value stayed at two. The synonym task 'verb: *hear*, object: *sound*' now has a higher number of votes than the user-defined task and will therefore take the place of the original user-defined task.

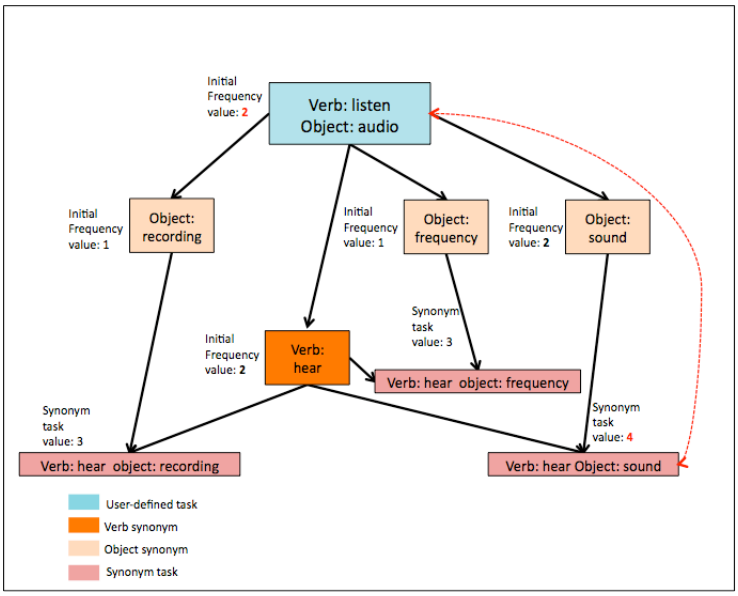

Figure 6.9: User-defined task and synonym task role switching.

6.4 Search Mechanism

The interface (Figure 6.10) provided for performing searches in the web template repository, allows to query the annotation database based on the parameters of title, tasks, user-defined tasks, device, theme, context and layout. It is also possible to restrict the number of results returned by entering the desired amount in the field provided for this purpose.

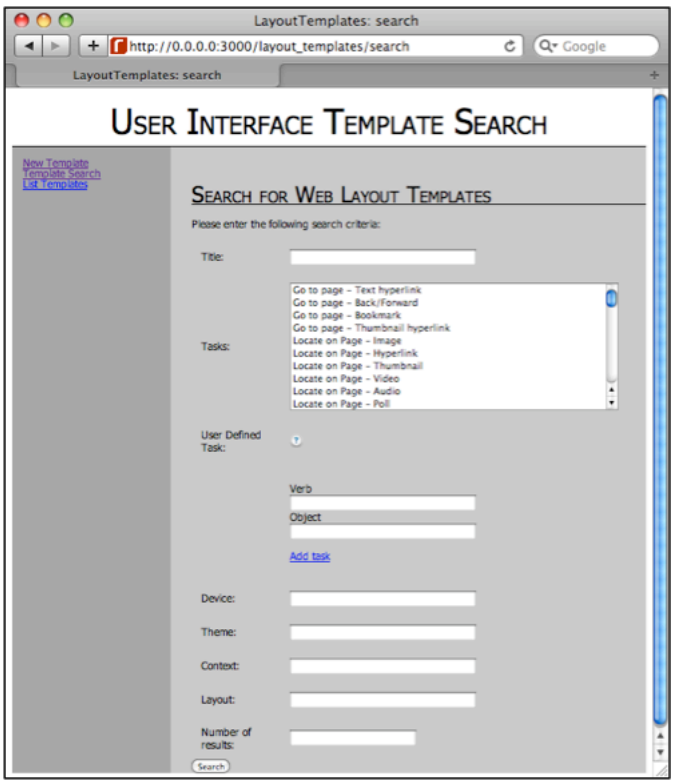

Figure 6.10: Web template search interface.

With the information that the user enters a query is created dynamically through the '*DynamicQuery*' class, seen in Figure 6.11. First, predefined query sentences are completed with the data entered in the search interface. Once all the parameters have been set, the '*constructQuery*' method is called, which determines which parameters were entered and which were not. This method also traverses through all the user-defined tasks that the user entered and completes the query by placing these values in query sentences that will execute a Boolean search on the user-defined task table. Finally, the '*constructQuery*' method concatenates all the necessary sentences to create the final query so it can be

executed.

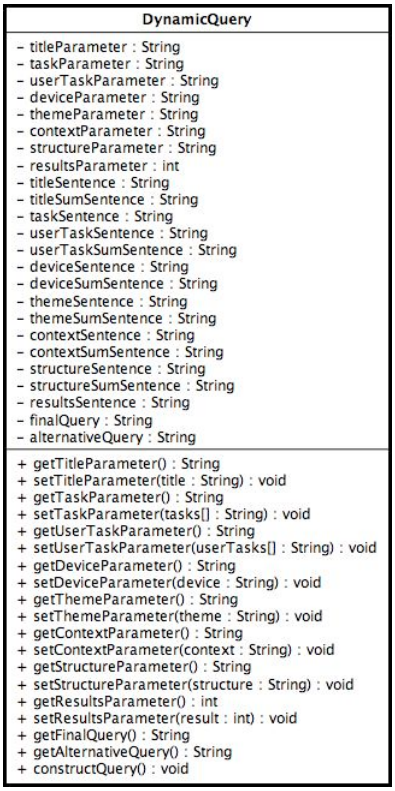

Figure 6.11: Dynamic query class.

The ranking of the results is computed within the query by doing a Boolean search on each of the parameters provided. With this function, if a match is found for a given parameter, the value of '1' is returned and '0' if there is no match. In the case of the user-defined task parameter, the value returned is additionally multiplied by the frequency it has times a factor of 0.5. In this way, the user-defined task is given a weight 50% higher than the rest of the parameters, because, as presented in Section 4.1, the functionality a web template can perform is the key aspect for the users who intend to reuse a template. Furthermore, by taking the frequency into consideration as well, the user voting effects the search results. For example, if two web templates are both described with the same user-defined task but one of them has more votes for that task than the other, it will rank higher, because several users believe this web template does in fact accomplish this task and there is a higher probability that this word is commonly understandable by the majority of users.

To compute the final score for each result, the values for each parameter are added

together. Hence the following formula:

*Score = sum(title + device + context + theme + sum(task) + (sum(user-defined task * (frequency)) * 1.5)*

In the case where no results match all the criteria that the user entered, then the synonyms are looked up to perform a new search with those terms instead and the formula above is also used to rank the results that this new search yields.

6.5 Operations

According to the objectives presented in Chapter 1, the implemented system should enable the users to add new web templates to the repository with its corresponding annotations. In order to increase the accuracy of the user task descriptions, the users should also be able to add more user task descriptions to existing web templates and new synonyms to enhance and disambiguate the description of their descriptions. Furthermore, the users should also have the ability to rank descriptions and synonyms given by other users. These implemented operations are presented in the following subsections.

6.5.1 Add New Web Template

When a user inserts a new web template into the repository, the template shall also be annotated to enable it to be accurately found by others. The template should also include a screenshot of the template, in order to allow the users to visually corroborate that they are accessing the right template according to their needs during future searches. Therefore the following steps take place when adding a new web template to the repository.

1. **Data entry.** System presents the fields of ID, screenshot, template file, title, device, theme, context, structure, task and user-defined task.
2. **Synonym search.** When the user adds a task description of their own to the web template, the system will search for synonyms for such task and present them to the user.
3. **Synonym selection.** It is the responsibility of the user to select the appropriate synonyms for the user task description entered.
4. **Upload.** Once the user has entered all the necessary information, the system will save the web template along with all the annotation data into the repository.

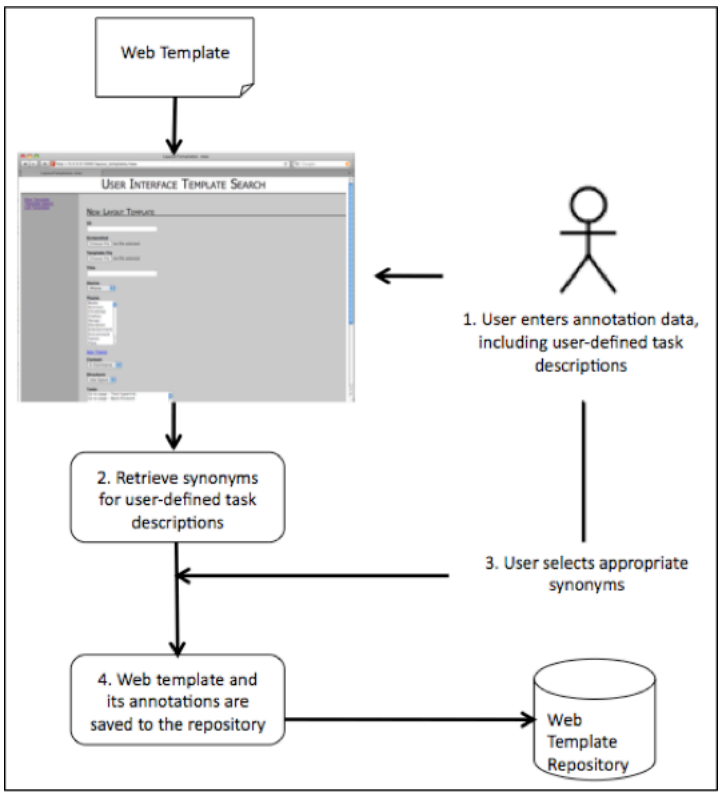

Figure 6.12: Adding a new web template to the repository.

6.5.2 Editing an Existing Web Template

In order to allow users to change the annotations either for the purpose of correcting errors or to add more user-defined tasks to an existing web template, the following steps take succession.

1. **Locate desired web template.** The user shall locate the desired web template, either through the search function or by browsing the list of all existing templates. The system displays all the annotation data for the selected web template and allows the editing of all fields.
2. **Add new user-defined tasks.** In the case of the user-defined task, the user is also presented with the opportunity to append new annotations. The system will provide a series of synonyms for the user-defined task that the user entered.
3. **Add synonyms.** The user now selects the synonyms that apply to the user-defined task previously entered.

4. **Save changes.** After the user has finished editing the web template, the system stores the changes to the annotations in the repository.

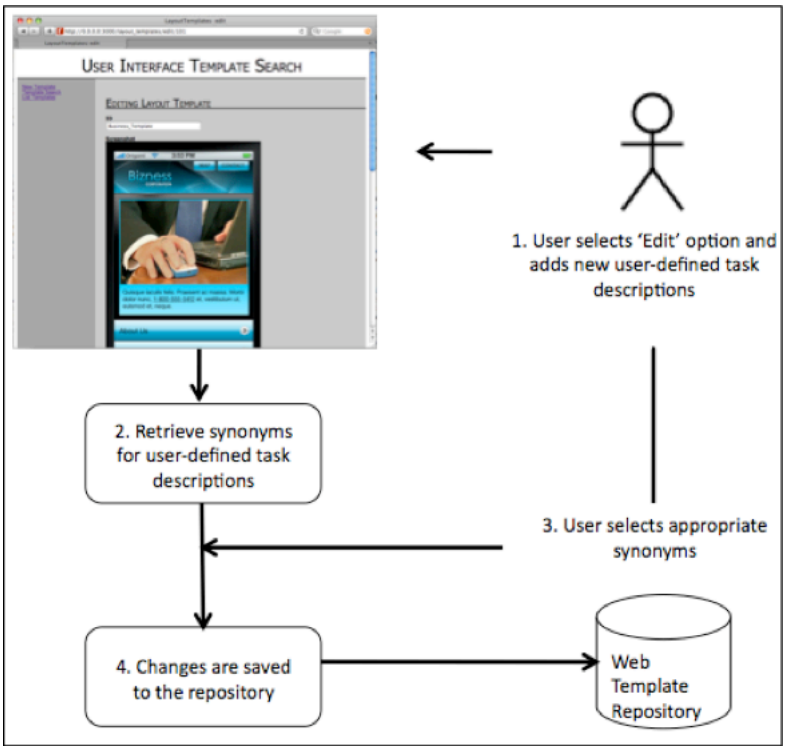

Figure 6.13: Editing an existing web template.

6.5.3 Voting on an Existing Web Template

To denote the relevance of a user-defined task or synonym and how they relate to a specific web template, the existing user-defined tasks and synonyms for a template can be voted on in the following way.

1. **Locate desired web template.** The user shall locate the desired web template, either through the search function or by browsing the list of all existing templates. The system displays all the annotations concerning the selected web template along with a screenshot of it. The user task descriptions, each one with their corresponding synonyms, are displayed with a check box next to them.
2. **Vote on user-defined tasks and synonyms.** The user selects what he or she considers are the user-defined tasks and/or synonyms that truly represent the functionality of the selected web template, and saves the choices made.

3. **Frequency check.** The system now saves the number of times, or frequency, that a user-defined task or synonym has been selected. With the new frequencies in place, the systems check to see whether the frequency of any the synonyms are higher than the frequency of its parent user-defined task.
4. **Relevance calculation.** If the frequency of any of the synonyms is higher than the frequency of the user-defined task, then the synonym becomes a user-defined task and the user-defined task is turned into a synonym because the users of the system are more likely to search for this web template using the terms that were originally synonyms.

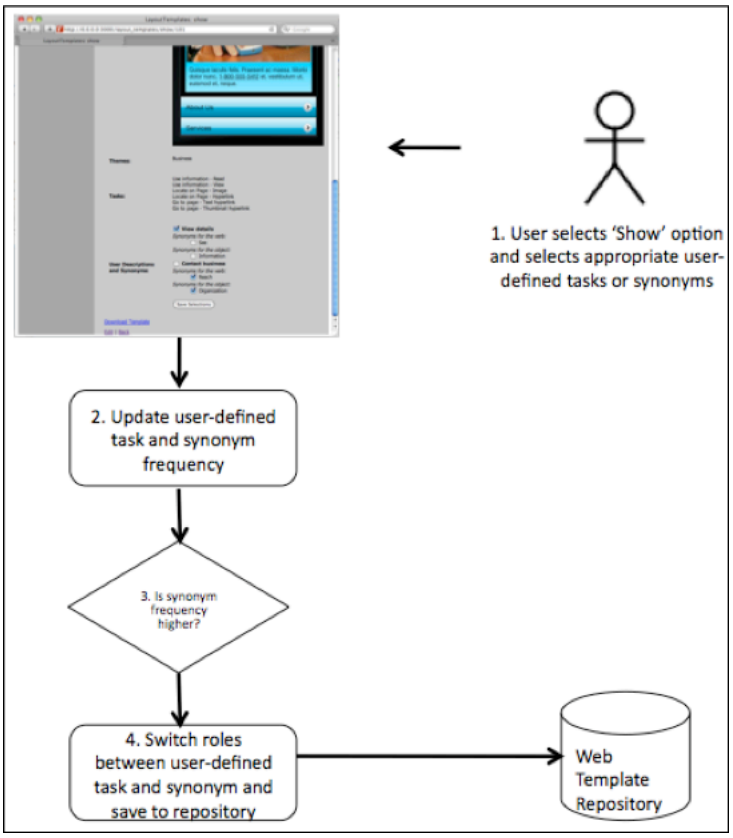

Figure 6.14: Voting on an existing web template.

6.5.4 Search Web Template

The following are the steps to take in order to give users the right web template according to their needs.

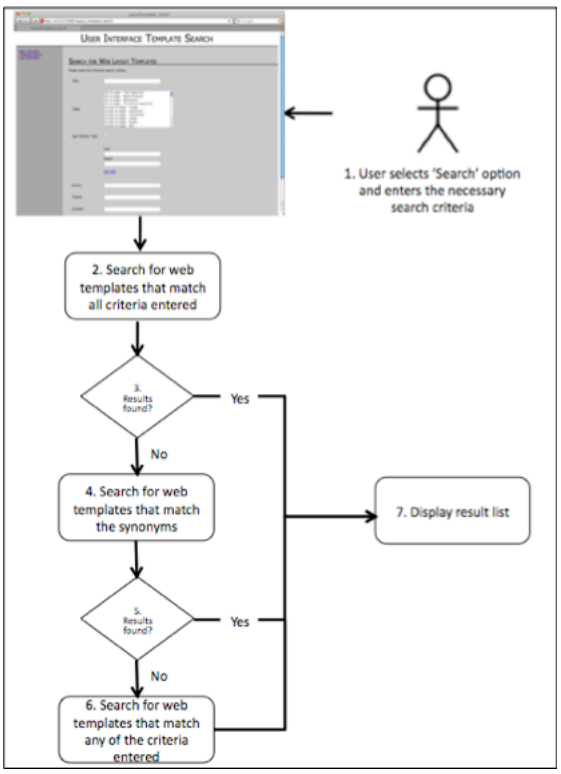

Figure 6.15: Searching for a web template.

1. **Enter search criteria.** The system will allow the user to enter the search criteria for the title, task, user-defined task, device, theme, context and structure. It will also allow the user to state the number of results to be returned.
2. **Perform search.** With the entered criteria, the system will search first for the web templates that meet all the requirements entered. Out of the matches found, a scoring mechanism will determine which one of the web templates more accurately matches the request of the user by taking into consideration the frequency with which its user tasks descriptions have been selected.
3. **Check results.** If no results are returned, then perform a synonym-based search (step 4). Otherwise, display the results (step 7).
4. **Synonym-based search.** The system will search for templates that meet all the criteria, but with the difference that it will search for synonyms for the user-defined task that the user initially entered and perform the search with these synonyms instead. The scoring mechanism will once again determine which web template more accurately meets the request.

5. **Check results.** If no results are returned, then perform an alternate search (step 6). Otherwise, display the results (step 7).
6. **Alternate search.** In case that no matches are found after the two previous searches, then the system will search for web templates that meet any of the initially entered criteria.
7. **Display results.** Finally, the system displays the results and the user is able to view or edit the returned web templates.

7 Evaluation

As explained in Section 4.1, in order to allow the reuse of web templates, annotations should be used to describe them and, most importantly, the tasks they are able to perform. The annotations of a web template represent the attributes by which the users will perform searches and the implementation of our approach provides an annotation system to describe the functionality of a web template. In addition our approach provides a search engine, which is specific for web templates, where the user is provided with a form where he or she can enter the desired criteria for each specific annotation property. On the other hand, web search engines, usually provide only one field to enter a Boolean search query. After looking at three of the most popular web search engines, namely Google[1], Yahoo![2] and Bing[3], we found that, although further details can be specified for the search through an '*Advanced search*' feature, these details usually entail limiting the results to a web domain, a certain geographical location or to a specific language, as well as allowing the user a to specify a Boolean search in a user-friendly graphical way.

Thus, we believe that, by annotating web template attributes, allowing the user to specify search criteria for each particular annotation property and ranking the results by giving a higher weight to the task descriptions, our search mechanism is more effective for our subject-matter and should yield more accurate results in terms of web template retrieval. Therefore, a comparison will be made between the results of similar queries executed on our system and on a web search engine.

This chapter presents a description of the experiments to be performed. Section 7.1 provides an overview of the evaluation methodology as well as the type of experiments to conduct.

7.1 Evaluation Methodology

Evaluations in terms of quality and performance will be carried out on the system with the goal of validating the effectiveness of the retrieval of web templates. In order to perform the experiments, we have chosen to evaluate our system against Google because

[1] http://www.google.com
[2] http://www.yahoo.com
[3] http://www.bing.com

it is one of the most widely used search engines and it provides Application Programming Interfaces (API) into their services, which should allow us to create customized searches to execute the necessary experiments. Given that Google bases its searches on the content of a web page, the annotations of the web templates in the repository will be published on one web page per web template. That is, one web page would display the annotation parameters and annotation values for one specific web template. Furthermore, the Google Custom Search will be configured to include only the web pages in the domain in which the annotations will be published, so as to avoid results from web pages external to ours.

The experiments will be carried out on a set of fifty different web templates in order to have diversity of functionality represented in those templates. This set of templates will be annotated and submitted to the repository by five different people to ensure we have varied vocabulary. In addition, five different queries will be entered into the system and the following measurements will be taken:

- Result count.
- Result order.
- Accuracy of results.
- Number of parameters that match search criteria.
- Query execution time.

The same five queries will be executed two times through a Google Custom Search API[4] and the same measurements will be taken as they were for our web template search system. The first cycle of query execution will entail entering the search criteria into the search field; however, in the second cycle the annotation properties will also be entered in order to see if the results differ when specifying to what each search parameter refers to.

The queries will be the following:

1. Search for a template that meets the criteria for all of the properties of a template, that is, title, task, one user-defined task, device, theme, context and layout.
2. Search for a template that meets the criteria for more than one user-defined task and the rest of the properties of a template, that is, title, task, device, theme, context and layout.
3. Search for a template that meets the criteria for one user-defined task parameter.
4. Search for a template that meets the criteria for more than one user-defined task parameter.
5. Search for a template that meets the criteria for all of the properties of a template, that is, title, task, one user-defined task, device, theme, context and layout and measure the response time.

[4] http://code.google.com/apis/customsearch/

In order to record the experiments, a questionnaire will be filled out, which can be found in Appendix A.

The results noted in the measurements will be compared between the two approaches to evaluate which one provides the better results in terms of result quality and response time.

In previous evaluations made on the system during and after the development phase in order to test the implementation of our approach, it was noted that the system returns results appropriately, according to the searches entered. The results are also displayed quickly, but it is possible that the response time of queries could increase with a larger repository. It was also noted that when there are no web templates that meet all of the requirements, the suggested results list might be too large and yield a lot of irrelevant results to the user in the cases where the results only meet one criterion.

8 Conclusion

The present work dealt with the annotation and retrieval process of web templates in order to encourage and simplify their reuse. First of all, a description of web templates and the tasks they are able to perform were described, as well as how lexical databases aid by supplying synonyms that can be applied to those tasks. A web template is a visual model that is integrated by all the graphical elements of a web page, whereas tasks represent the functionality that can be performed by those web templates. A web template can be identified and distinguished from others by describing its functionality and a good description can be complemented by specifying synonyms for it, which can be retrieved through a lexical database.

Furthermore, a Template Model was introduced, which encompasses the attributes that can be used to accurately describe a web template, which are, tasks, user-defined tasks, synonyms, theme, context, device, structure, widgets and position. These attributes serve to distinguish one web template from another by providing details of their characteristics and, given the fact that these attributes are associated to a web template in the form of annotations, users can search for templates based on these properties.

In order to find the templates that have been annotated in a repository, a Web Template Retrieval strategy was proposed, in which the user-defined tasks have a higher priority over the rest of the annotation properties because the user-defined tasks represent the functionality that the users need, according to the application that is to be developed. In addition, the user-defined tasks receive a higher priority because they also represent the vocabulary most likely to be used when performing searches. In addition, the use of synonyms as a way to express equivalent tasks was presented with the help of the lexical database WordNet as a source. The more alternatives that are provided to describe a task, the more likely it is for users to find the right template according to their needs, that is why the use of synonyms is included in the system.

Furthermore, a voting mechanism was put in place in order to allow users of the system to influence the relevance of the user-defined tasks and synonyms, which makes it possible for users to choose, through a majority consensus, the vocabulary that best describes the functionality of a web template, which in consequence, is what the users will most probably use later on when they search for web templates. Given that the user-defined task is the vocabulary that is most likely to be used when performing a search in the system, if a synonym is voted as a better task description than a user-defined task, then

the system is able to change the role of the synonym to that of a user-defined task and relegate the original user-defined task to a synonym.

In order to allow the users to find the web templates they need, a search mechanism was established, which returns ranked results and gives higher priority to web templates that meet the criteria in terms of the tasks that it is capable of performing. The search mechanism also takes the voting on user-defined tasks and synonyms into consideration in order to be able to determine which web template fits a task better than another, because if a web template has more votes than another on the same user-defined task, it means that more people believe that web template fits the description better.

While the annotations of a web template provide a series of details of a web template to make it identifiable in a search, the system does not allow for synonyms on any other attribute other than the user-defined task. Also in terms of synonyms, the system does not contemplate the user-defined task as an expression but, instead, as two words: a verb and an object.

The summarization of the conclusions and contributions of this work is the following:

- In order to be able to retrieve a web template from a repository, annotations of its features are required. Therefore, a template model that provides a detailed description of the functional and visual features of a web template was created.
- Given that users can describe a template in many different ways, a mechanism to determine the vocabulary most likely to be used to describe a web template task within a group of users was established.
- A search mechanism that prioritizes the tasks a web template is able to perform was created, because the reuse of a web template is mainly prompted by the need of a user interface that conforms to specific functional requirements.

8.1 Future Work

This section discusses the enhancements that can be done to improve the system.

8.1.1 Synonym Task Enhancement

An approach to accurately describe the functionality of a web template can be achieved by annotating the tasks it can perform. Additionally, creating synonym tasks can further enhance the annotations by providing alternative vocabulary by which the tasks can be used. Currently the system treats synonym tasks as two separate words, so one solution to improve them could be to evaluate the whole expression of a user-defined task and create an equivalent synonym expression from it. This way the synonym of the task can more accurately capture the meaning of the task that the user is describing.

8.1.2 Search Based on Structural Elements

Web template annotations describe the characteristics of a web template and therefore allow users and machines alike to be able to distinguish them from one another when doing searches. In this work, the description of the structure of a web template has been considered, along with the elements it holds and the position they have within it. However, these characteristics are not integrated into the search mechanism. The structural annotations could be further explored as to what kind of information the widgets can provide in order to refine search queries.

An approach to providing an enhanced description of the structural properties of a web template is by creating a model of all the possible elements of a template and how they relate between one another. Moreover, research can be done to find to find the trends of the structural properties of web templates, which could attempt to demonstrate if there is a relationship between the functionality of a template and the specific combination of widgets in a certain location within that template.

A Appendix

A.1 Evaluation Questionnaire

1. Search for a template that meets the criteria for all of the properties of a template, that is, title, task, one user-defined task, device, theme, context and layout.
    a) Enter your search criteria:
        i. Title:
        ii. Task:
        iii. User-defined task:
        iv. Device:
        v. Theme:
        vi. Context:
        vii. Layout:
    b) How many results were returned?
    c) What are the ID numbers of the first five results returned?
    d) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?
        i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?
    e) How many parameters matched the search criteria on the first result returned?
2. Perform a search with the parameters in 1 in Google.
    a) How many results were returned?
    b) What are the ID numbers of the first five results returned?
    c) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?

   i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?

d) How many parameters matched the search criteria on the first result returned?

3. Perform a search with the parameters in 1 in Google, but also include the annotation properties (title, task, user-defined task, device, theme, context, layout) in the search.

   a) How many results were returned?

   b) What are the ID numbers of the first five results returned?

   c) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?

      i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?

   d) How many parameters matched the search criteria on the first result returned?

4. Search for a template that meets the criteria for more than one user-defined task and the rest of the properties of a template, that is, title, task, device, theme, context and layout.

   a) Enter your search criteria:

      i. Title:

      ii. Task:

      iii. User-defined tasks:

      iv. Device:

      v. Theme:

      vi. Context:

      vii. Layout:

   b) How many results were returned?

   c) What are the ID numbers of the first five results returned?

   d) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?

      i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?

   e) How many parameters matched the search criteria on the first result returned?

5. Perform a search with the parameters in 4 in Google.
    a) How many results were returned?
    b) What are the ID numbers of the first five results returned?
    c) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?
        i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?
    d) How many parameters matched the search criteria on the first result returned?
6. Perform a search with the parameters in 4 in Google, but also include the annotation properties (title, task, user-defined task, device, theme, context, layout) in the search.
    a) How many results were returned?
    b) What are the ID numbers of the first five results returned?
    c) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?
        i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?
    d) How many parameters matched the search criteria on the first result returned?
7. Search for a template that meets the criteria for one user-defined task parameter.
    a) Enter your search criteria:
        i. User-defined task:
    b) How many results were returned?
    c) What are the ID numbers of the first five results returned?
    d) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?
        i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?
    e) How many parameters matched the search criteria on the first result returned?
8. Perform a search with the parameters in 7 in Google.
    a) How many results were returned?

b) What are the ID numbers of the first five results returned?

c) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?

i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?

d) How many parameters matched the search criteria on the first result returned?

9. Perform a search with the parameters in 7 in Google, but also include the annotation property (user-defined task) in the search.

a) How many results were returned?

b) What are the ID numbers of the first five results returned?

c) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?

i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?

d) How many parameters matched the search criteria on the first result returned?

10. Search for a template that meets the criteria for more than one user-defined task parameter.

a) Enter your search criteria:

i. User-defined tasks:

b) How many results were returned?

c) What are the ID numbers of the first five results returned?

d) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?

i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?

e) How many parameters matched the search criteria on the first result returned?

11. Perform a search with the parameters in 10 in Google.

a) How many results were returned?

b) What are the ID numbers of the first five results returned?

c) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?

i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?

d) How many parameters matched the search criteria on the first result returned?

12. Perform a search with the parameters in 10 in Google, but also include the annotation property (user-defined task) in the search.

a) How many results were returned?

b) What are the ID numbers of the first five results returned?

c) In a scale of 1 to 10 with 10 being the highest, how accurate is the first result of the search?

i. If the above score was 5 or less, was there another result that did deserve a score higher than 5? In what place was this result returned?

d) How many parameters matched the search criteria on the first result returned?

13. Search for a template that meets the criteria for more than one user-defined task and the rest of the properties of a template, that is, title, task, device, theme, context and layout and measure the time it takes for the search mechanism to return results.

a) Enter your search criteria:

i. Title:

ii. Task:

iii. User-defined tasks:

iv. Device:

v. Theme:

vi. Context:

vii. Layout:

b) How much time did it take the system to display the results?

14. Perform a search with the parameters in 13 in Google.

a) How much time did it take Google display the results?

15. Perform a search with the parameters in 13 in Google, but also include the annotation (title, task, user-defined task, device, theme, context, layout) in the search.

a) How much time did it take Google display the results?

Bibliography

[ABB+99] Antonella Andreoni, Maria B Baldacci, Stefania Biagioni, Carlo Carlesi, Donatella Castelli, Pasquale Pagano, Carol Peters, and Serena Pisani. The ercim technical reference digital library: Meeting the requirements of a european community within an international federation. Technical report, 1999.

[Bec04] Dave Beckett. Rdf/xml syntax specification, 2 2004.

[BJWC99] Michael D. Byrne, Bonnie E. John, Neil S. Wehrle, and David C. Crow. The tangled web we wove: A taskonomy of www use. pages 544–551. ACM Press, 1999.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. In *Computer Networks and ISDN Systems*, pages 107–117, 1998.

[Bro02] Andrei Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[Cor03] Paragon Corporation. What do we mean by separation of business logic from presentation logic? 2003.

[DEFS06] Pavel A. Dmitriev, Nadav Eiron, Marcus Fontoura, and Eugene Shekita. Using annotations in enterprise search. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 811–817, New York, NY, USA, 2006. ACM.

[DFJ+04] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, R. Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. Swoogle: a search and metadata engine for the semantic web. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 652–659, New York, NY, USA, 2004. ACM.

[DG04] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. pages 137–150, 2004.

[DOS03] Michael C. Daconta, Leo J. Obrst, and Kevin T. Smith. *The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management*. Wiley Publishing, Inc., 2003.

[DS04] Dan Diaper and Neville A. Stanton. *The Handbook of Task Analysis for Human-Computer Interaction*. Lawrence Erlbaum Associates, Inc., Publishers, New Jersey, 2004.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, illustrated edition edition, November 1994.

[GPT05] David Gibson, Kunal Punera, and Andrew Tomkins. The volume and evolution of web page templates. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 830–839, New York, NY, USA, 2005. ACM.

[Gra09] Hagen Graf. *Joomla! 1.5 Websites organisieren und gestalten mit dem Open Source-CMS*. Addison Wesley, 2009.

[Her09] Ivan Herman. Semantic web, 11 2009.

[HS84] Christopher A. Higgins and Frank R. Safayeni. A critical appraisal of task taxonomies as a tool for studying office activities. *ACM Trans. Inf. Syst.*, 2(4):331–339, 1984.

[JR09] Yushi Jing and Henry Rowley. Explore images with google image swirl, November 2009.

[JS01] M. Jarmasz and S. Szpakowicz. Roget's thesaurus: a lexical resource to treasure. In *Proceedings of the NAACL WordNet and Other Lexical Resources workshop*, pages 186–188, Pittsburgh, June 2001.

[JS03] Mario Jarmasz and Stan Szpakowicz. Roget's thesaurus and semantic similarity. In *Conference on Recent Advances in Natural Language Processing*, pages 212–219, 2003.

[Ken00] Peter Kentie. *Web graphics - Tools und Techniken für die Web-Gestaltung*. Addison Wesley Longman bei Pearson Education Deutschland, München, 3rd edition, 2000.

[KS08] Alistair Kennedy and Stan Szpakowicz. Evaluating Roget's thesauri. In *Proceedings of ACL-08: HLT*, pages 416–424, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[LL06] Christoph Lindemann and Lars Littig. Coarse-grained classification of web sites by their structural properties. In *WIDM '06: Proceedings of the 8th annual ACM international workshop on Web information and data management*, pages 35–42, New York, NY, USA, 2006. ACM.

[MF99] Katsushi Matsuda and Toshikazu Fukushima. Task-oriented world wide web retrieval by document type classification. In *CIKM '99: Proceedings of the*

eighth international conference on Information and knowledge management, pages 109–113, New York, NY, USA, 1999. ACM.

[Mil95] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, November 1995.

[MPC01] Julie B. Morrison, Peter Pirolli, and Stuart K. Card. A taxonomic analysis of what world wide web activities significantly impact people's decisions and actions. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 163–164, New York, NY, USA, 2001. ACM.

[MR92] B A Myers and M B Rosson. Survey on user interface programming. In *In SIGCHI'92: Human Factors in Computing Systems*, 1992.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. Owl web ontology overview, 2 2004.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[QD09] Xiaoguang Qi and Brian D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41(2):1–31, 2009.

[RG02] Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems*. McGraw-Hill Science/Engineering/Math, 3 edition, August 2002.

[Rog91] Peter Mark Roget. *Roget's Thesaurus of English Words and Phrases*. Penguin, 1991.

[SFB06] L. Christian Schaupp, Weiguo Fan, and France Belanger. Determining success for different website goals. In *HICSS*. IEEE Computer Society, 2006.

[SKV08] Vassilis Spiliopoulos, Konstantinos Kotis, and George A. Vouros. Semantic retrieval and ranking of semantic web documents using free-form queries. *Int. J. Metadata Semant. Ontologies*, 3(2):95–108, 2008.

[UG96] Mike Uschold and Michael Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11(2):93–155, 1996.

[Vos98] Piek Vossen. Introduction to eurowordnet. pages 1–17, 1998.

[Wei93] Mark Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36(7):75–84, 1993.

Selbständigkeitserklärung

Hiermit erkläre ich, Paoli Izquierdo Llanes, die vorliegende Masterarbeit zum Thema *Retrieval of User Interface Templates Based on Tasks* selbstständig und ausschließlich unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst und Zitate kenntlich gemacht zu haben.

Dresden, den 15.1.2010