

Structuring Meta-search Research by Design Patterns¹

Jürgen Dorn and Tabbasum Naz
Institute of Information Systems 184/2, Technical University Vienna,
Favoritenstraße 9-11, Vienna, A-1040, Austria
{dorn|naz}@dbai.tuwien.ac.at

Abstract

Design patterns shall support the reuse of a software architecture in different application domains as well as the flexible reuse of components. In this paper, we propose design patterns for meta-search engines. We also introduce design patterns for common components of meta-search engines e.g. query interface generator, information extraction, result merger and result ranker. Presented design patterns for meta-search engines and their components are reusable, extendable and flexible. These design patterns accelerate the development process in meta-search domain and other related domains. Moreover, it promises higher quality of developed solutions. These design patterns also provide developers with a shared vocabulary for easy communication.

Keywords: meta-search, specialized search engine, design patterns

1. Introduction

Internet is flooded with information and contains hundreds of Web sites with thousands of topics in which searchers get lost while searching a topic. Second most popular activity in the Internet after e-mail is “search” [1]. One Microsoft researcher says: “*Estimates are that information workers spend as much as 30 percent of their time searching for information, at a cost of \$18,000 each year per employee in lost productivity*” [2]. Primary search tools i.e. search engines, subject directories, social network search engines are available for the searchers but these are not sufficient to meet the requirements of users and are unable to provide the desired results. They have limited coverage, cannot locate high quality information from the invisible Web stored in accessible databases because of technical limitations or because of exclusion from the indices and the returned search results consist of long documents. Searchers have to navigate through long documents to find the relevant information from these long documents. One of Microsoft reports says: “*people search an average of 11 minutes before they find what they are looking for*” [3]. Subject directories are organized on the Internet sites by subject and users choose a subject of interest from the list of subjects. Subject directories depend on human editors for listings. If the description of a site is not specific enough then search may be unsuccessful. Social network services allow people with shared interests, hobbies, or causes to come together online. Social network search engines (del.icio.us, digg, reddit) are a class of search engines that use social networks to organize, prioritize, or filter search results. Social bookmark sites, allow

¹ Published at International Computer Science and Technology Conference, San Diego, April 1st-3rd, 2008, received best paper award

Internet users to share content they like best with others for searching and viewing. Users can assign tags to their favourite Web pages. But the problem with such type of tag-based systems is that there is no set of controlled vocabulary for tags and no standard for tag structures. There also exist spelling errors and multiple meanings of tags. Some users are also misusing tags to make their Web sites more visible [4].

To overcome all these problems with traditional search tools, “meta-search engine” are proposed as an choice for specific topic search. Meta-search engines (MSE) also known as multi-threaded engines, do not necessarily maintain their own listings/databases, but send the user’s query simultaneously to other search engines, Web directories or to deep Web, collect the results, remove the duplicate links, merge and rank them according to their own algorithm in a single list and display it to the user. Meta-search engines provide fast and easy access to the desired search, because they can search from multiple search engines simultaneously and save the precious time of the searcher. Meta-search engines provide a broader overview of a topic as compared to traditional search engines and increase the coverage of Web by combining the coverage of multiple search engines. Querying multiple search engines is more scalable then the centralized general purpose search engine. Meta-search engines have the ability to search the invisible Web too thus increasing the precision, recall, and quality of results. They make the user task much easier by searching and ranking the results from multiple search engines [5].

A lot of research is in progress in the field of developing configurable meta-search engines in different domains i.e. jobs, hotels, flights, news, research papers and real estate etc. It has been observed that developing a configurable meta-search engine in any domain is a tedious and time consuming task. Every time developers have to start the development process from scratch. It is desirable to have a reusable and flexible design for meta-search engines. After a detailed study of meta-search engine development research, we identified different processes and components for meta-search engines that meet specific requirements. In this paper, we propose reusable and flexible design patterns for meta-search engines and their components so that they can be reused in several times after some modifications. The design patterns of meta-search engine i.e. result ranker can also be reused in other application domains .

The rest of the paper is organized as follows. Section 2 describes the related work in the development of meta-search engines, design patterns and frameworks for domain specific search engines from a design pattern perspective. The discussion of different solutions motivate also the rationale for the design pattern. Section 3 provides an overall architecture for meta-search engines . Section 4 contains design patterns for meta-search construction and usage as well as important common components of meta-search engines. Finally, sections 5 concludes our work.

2. Related Work

In object-oriented systems there exist recurring patterns of classes and communicating objects. These patterns provide simple and elegant solutions to specific design problems and make object-oriented designs more flexible. In [6], Gof (Gang of four) describe 23 of the most common patterns in detail. These design patterns can help developers to structure their own specific applications and give them a common vocabulary to describe design concepts, rather than particular implementations [6][7].

To our knowledge, [8] is the only research that describes the framework for domain specific search engines from design patterns perspective. [8] also present design patterns for some components of domain specific search engines.

The Web Database Metasearch Engine project is developing technologies for providing integrated access to Web databases. Important phases for interface integration are automatic schema matching, schema integration and data integration. [9] use meta-information from Web search interfaces and present a two-step clustering based approach i.e. positive match based clustering and predictive match based clustering for schema matching.

MetaQuerier project (<http://metaquerier.cs.uiuc.edu/>) explores and integrates the query databases that are not visible to the traditional crawlers. Main components of MetaQuerier are MetaExplorer and MetaIntegrator. MetaExplorer discover sources on the deep Web and builds a search engine of Web databases. Moreover, MetaExplorer also creates models to represent discovered databases and develop wrappers to automatically extract schema details. MetaQuerier applies a holistic schema matching approach for schema matching. Holistic schema matching approach is used to identify simple 1:1 matching and complex 1:n or m:n matchings [10] [11].

In our previous research [12][13], we integrate job portals by meta-search and use a domain ontology for schema matching, schema integration, and data integration. The domain ontology is also used for information extraction from the result pages returned by various search engines. In [14] we have applied meta-search in the domain of accommodation search. Here the extraction from Web sources is based on Web service interfaces, requiring not all components that are required in the job domain. In a further project we address the search for orders in the logistics domain.

[15] introduce some techniques to automatically extract search result records (SRR) from dynamically generated HTML result pages. They present a tool ViNTs (Visual Information aNd Tag Structure) that utilizes visual content features and HTML tag structure of HTML result pages for the automatic wrapper generation of any given search engine. [16] presents techniques for supervised wrapper generation and automated Web information extraction. They implemented these techniques in a system called Lixto. Lixto provides a visual, interactive and convenient user interface for the creation of semi-automatic wrapper programs. The Lixto wrapper generator consists of modules i.e. navigator, extractor and visual developer. Lixto wrapper generator translates required piece of information from HTML pages into XML. [17] proposes an algorithm MDR (Mining Data Records in Web Pages) to mine data records in a Web page automatically. Authors claim that their data mining technique is able to mine both contiguous and non-contiguous data records.

[17] investigate result merging algorithms for meta-search engines, to merge results from different search engines into a single ranked list and state that merging based on titles and snippets of retrieved results can outperform other approaches based on full document analysis. They present five algorithms i.e. TopD, TopSRR, SRRSim, SRRRank and SRRSimMF for merging results into a single ranked list. Search result records (SRR) contain URL, title, and summary (snippet) of the extracted document. TopD algorithm uses the top document while

TopSRR algorithm use top SRRs to compute the search engine score. The SRRSim algorithm computes similarities between SRRs and query. SRRRank algorithm rank SRRs using more features like location of the occurred query term or total number of occurrences of the query term in the title and snippet etc. SRRSimMF algorithm computes similarities between SRRs and query using more features.

3. Meta-Search Architecture

This section describes the overall design of a meta-search engine. We identified that there are two processes in meta-search engines i) the meta-search engine creation process and ii) the meta-search engine usage process. Figure 1 and 2 shows the main components involved in the meta-search engine creation and usage process.

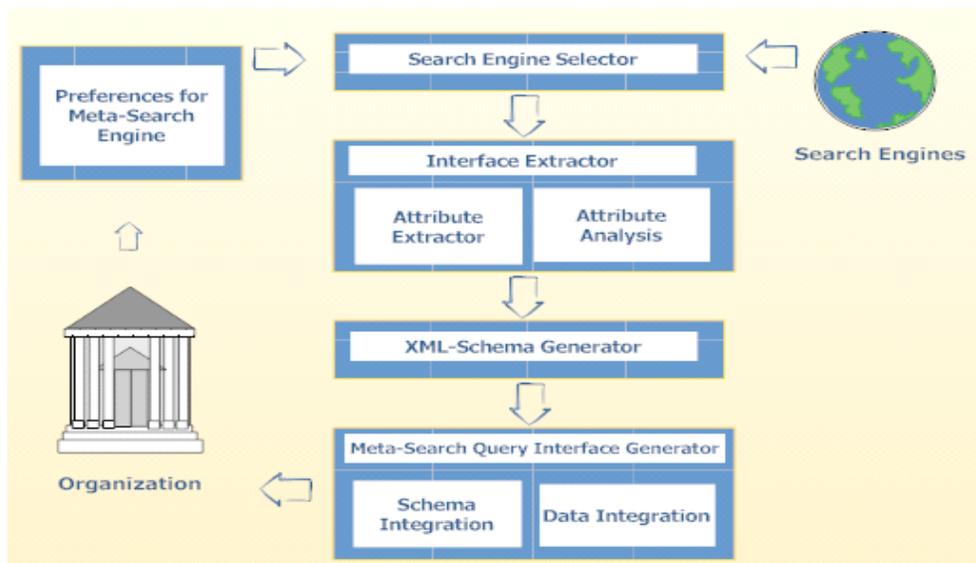


Fig. 1: Meta-Search Creation Process Components

Meta-search engine creation process work as follows. First of all, a developer specifies preferences like for which type, country or geographical area a meta-search engine is required by a *preferences collector* component. After getting preferences the *search engine selector* component will be activated and search engines meeting the preferences of the developer will be selected. Next, the *Interface extractor* component derives and analyse attributes from Web search interfaces. Then, an *XML Schema generator* component creates XML schemes for every search interface. Finally, a *query interface generator* component matches and integrates different XML-Schemes to have a single *query interface* for the meta-search engine.

When a user/seeker sends a query from the “meta-search query interface” to the meta-search engine, the components involved in meta-search engine usage process (Figure 2) will be activated and works as follows. Queries from the query interface are dispatched by a *query dispatcher* component and result pages with lists of results are collected. The *information extractor* component is responsible for extraction of records and the results from the result pages.

Next all the identified results are merged together by a *result merger* component and stored in a *database* or in *XML* format. Duplicate results are removed by a *duplicate result eliminator* component and finally results are ranked by a *result ranker* component.

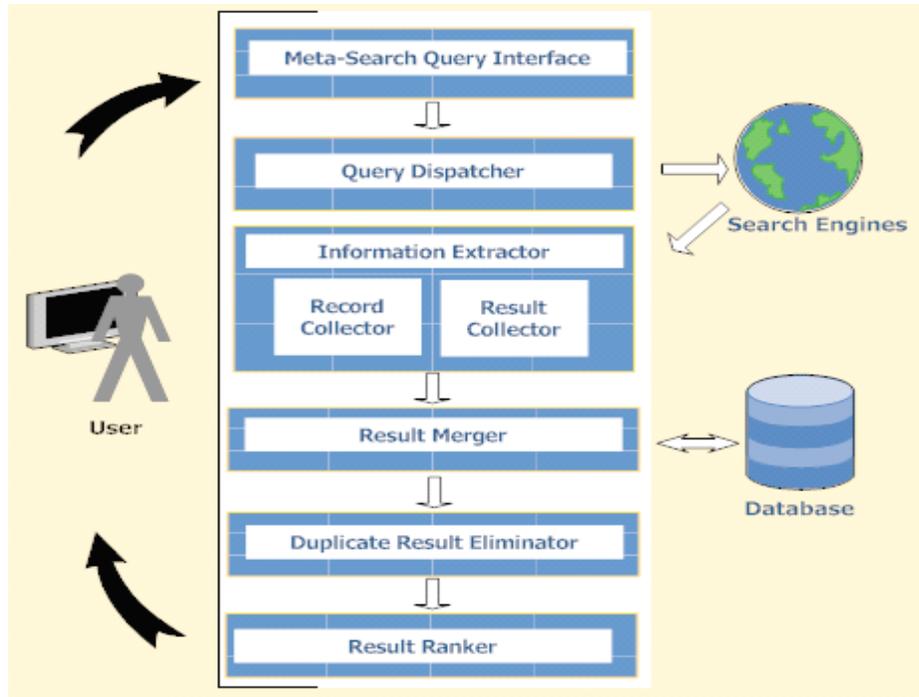


Fig. 2: Meta-Search Usage Process Components

4. Design Patterns for Meta-Search Engines

This section contains the designed patterns for both processes and the main components of meta-search engines. Figure 3 and 4 describes our meta-search processes by using an abstract factory design pattern. “*Abstract factory pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes. It defines the interface that all concrete factories must implement, which consists of a set of methods for producing products*” [6][7]. The meta-search abstract factory can produce any type of meta-search engine, as long as it gets proper set of directions, called factories. A factory design patten will build a certain type of meta-search engine, depending upon the type of factory. It is clear from the abstract factory pattern in Figure 3 and 4, that families of job meta-search objects or hotel meta-search objects can be created from a developer or a user perspective. Each type of meta-search engine has the same overall structure that all meta-search engines share in common i.e. interface extractor, XML-schema generator, query interface generator, information extractor, result merger, duplicate result eliminator and result ranker etc. New factories for flights or real estates search etc. can be added easily. Using the meta-search abstract factory pattern, we do not need to worry about what kind of meta-search we are building. An abstract class may contain a default method i.e. a simple ranking algorithm, that can be used by every concrete meta-search engine, but will be refined if more specific ranking is required. Job-MS-Factory in Figure 3 represents a Job-

Meta-Search-Creation-Factory and Job-MSU-Factory in Figure 4 represents Job-Meta-Search-Usage-Factory. Few parts in Figure 3 and 4 are not drawn for the sake of saving space.

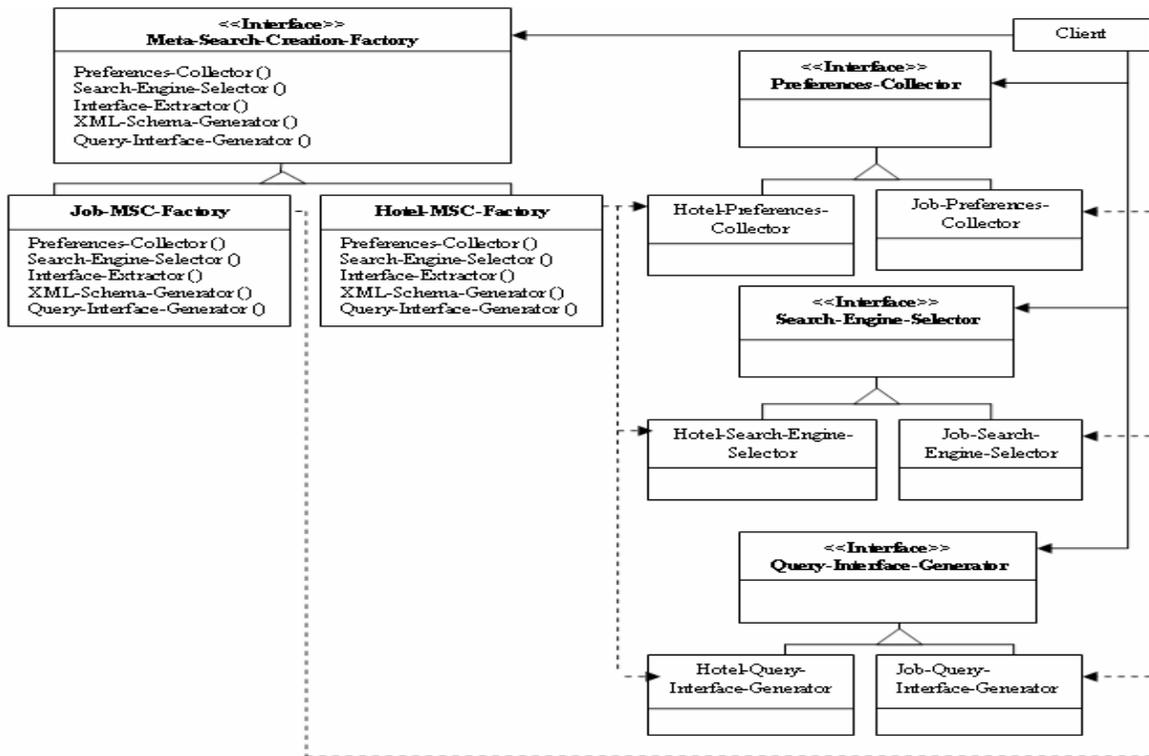


Fig. 3: Abstract Factory P attern for Meta -Search Engines Creation Process

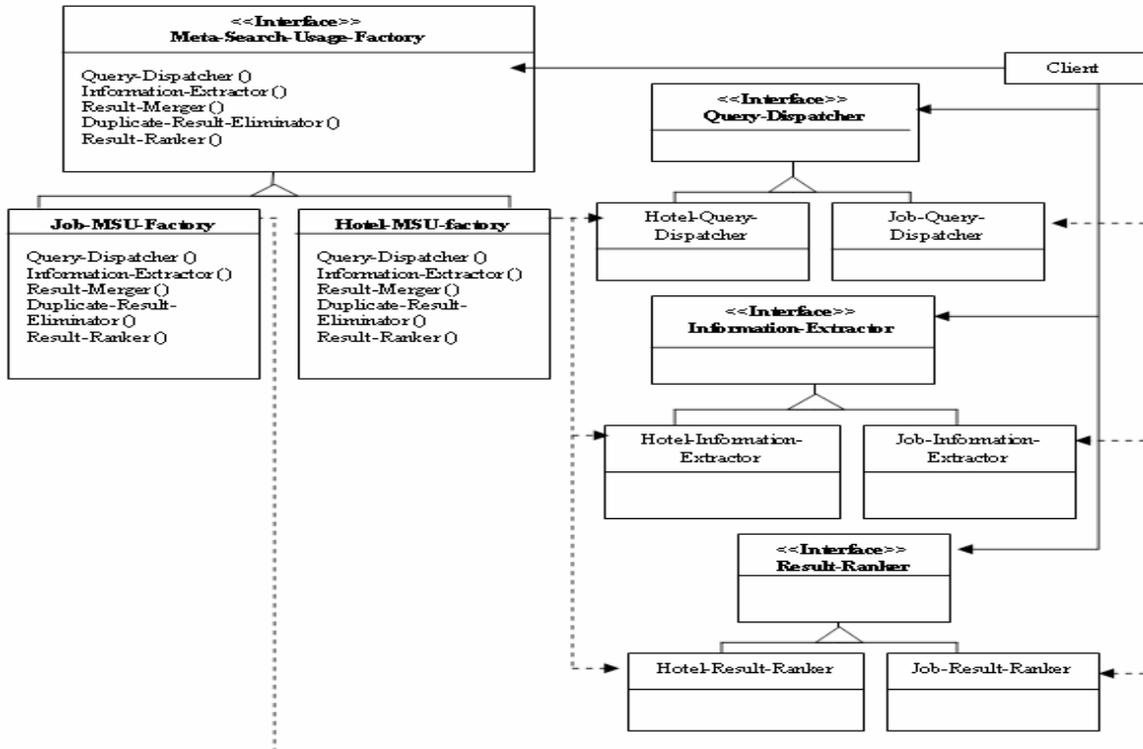


Fig. 4: Abstract Factory P attern for Meta -Search Engines Usage Process

Below are some design patterns for common and important components of meta-search engines.

4.1. Query Interface-Generator

The query interface generator component of a meta-search engine is responsible for schema integration, data integration and then production of query interface for meta-search engine. Integration of interface schemes is divided into two parts i.e. schema matching and schema merging. During schema matching, semantic correspondence between interface attributes is identified and each schema is translated into a single schema for the query interface. During data integration, the values of different attributes for the user interface are determined. It is required that values are semantically unique and compatible with the local values. Different methods have been proposed for schema and data integration by using i) domain ontology [12][13], ii) clustering approach [9] or iii) holistic schema matching approach [10][11]. Different approaches for schema matching meet specific requirements according to the specific context. So developers should have a facility to choose one of the above mentioned approaches according to the specific requirement. It is required that new algorithms for schema and data integration comply with the same interface. These algorithms can easily be introduced with less effort and without changing the other code.

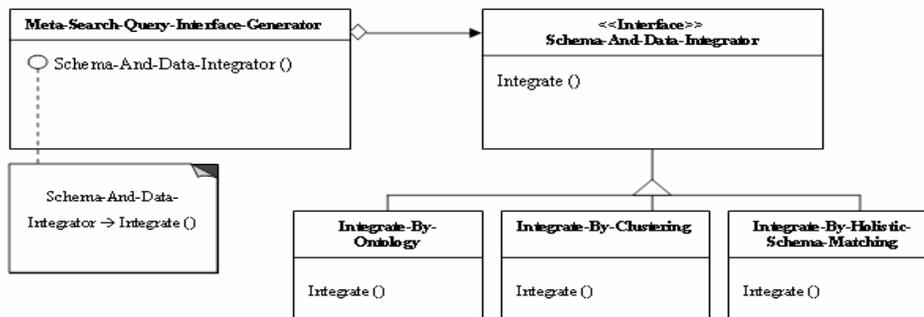


Fig. 5: Strategy Pattern for Meta -Search Query Interface Generator

To meet above requirements, the strategy design pattern as shown in Figure 5 is used for the query interface generator. “*Strategy design pattern defines a family of algorithms, encapsulate each one, and makes them interchangeable. Strategy lets the algorithm vary independently from clients that use it*” [7]. The design pattern in Figure 5 implements that the Meta-Search-Query-Interface-Generator is a class that is responsible for schema and data integration of different search engines and Schema-And-Data-Integrator is an interface. Integrating strategies is not implemented by the class Meta-Search-Query-Interface-Generator. Instead, they are implemented separately by sub-classes of abstract Schema-And-Data-Integrator class. Sub-classes of abstract Schema-And-Data-Integrator class implement different integrate strategies i.e. Integrate-By-Ontology, Integrate-By-Clustering and Integrate-By-Holistic-Schema-Matching. To switch schema and data integrator strategies, each meta-search engine calls the integrate method that it prefers. If a developer wants to add a new schema and data integration algorithm into the system, this can be done easily by implementing a new class using the Schema-And-Data-Integrator interface.

4.2. Information Extractor

The *information extractor* component is responsible for extraction of results from the result pages. It consists of a *Record collector* component and a *Result Collector* component. The *record collector* component is responsible for identification of the record section from the result page i.e. list of jobs, table with flights etc and *Result collector* component is responsible to

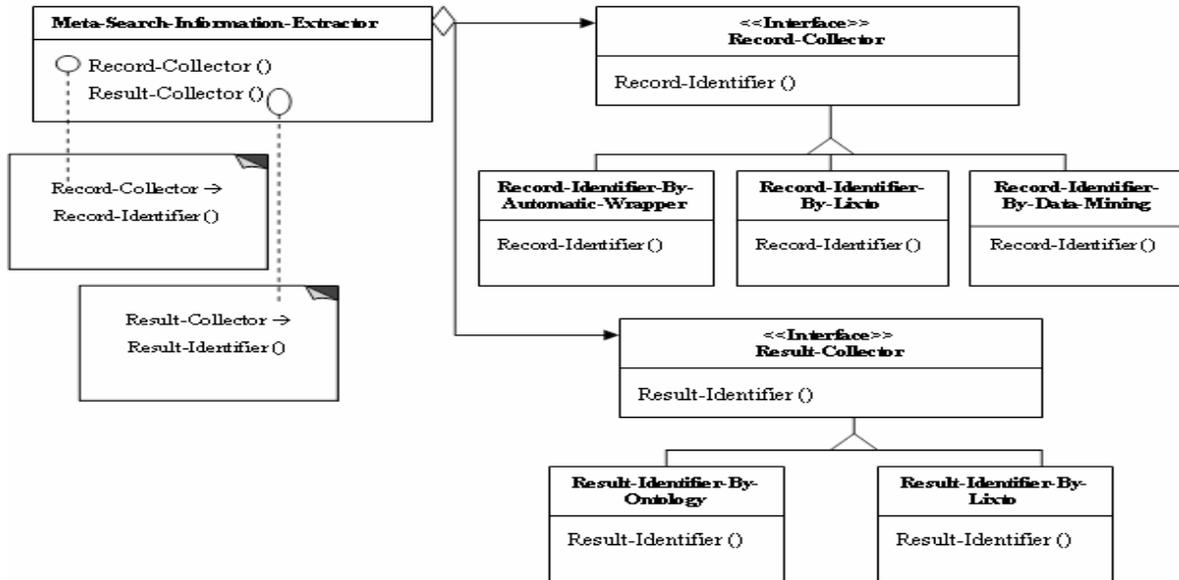


Fig. 6 : Strategy Pattern for Meta -Search Information Extractor

extract the exact fields i.e. job salary, hotel price etc, from the identified record section. Information extraction research shows that information extraction from different websites is often performed by using wrappers. Wrappers can be constructed manually, semi-automatically and automatically for record section identification. For identification of record section, different approaches i.e. i) automatic wrapper generation as ViNTs [14], ii) supervised wrapper generation as Lixto [15] or iii) data mining approaches [16] can be utilized. After identification of record section, a meta-search *result identifier* component is utilized for the extraction of exact results by using i) a domain ontology [12] [13] or ii) Lixto information extraction tool again [15].

For schema matching and merging it is required to normalize the terms like “Posted Date” into “Post Date” or “Job Types” to “Job Type”. Stemming algorithms i.e., Porter’s stemming algorithm can be utilized for term normalization process. A stemming algorithm is a method to convert word to their related form i.e. root, stem, or base. The stemming process is useful to find similar terms by only considering the word stem in search engines, natural language processing, and text processing.

The strategy design pattern is utilized for the information extraction component of the meta-search engine. The design pattern in Figure 6 shows that the Meta-Search-Information-Extraction class is responsible for information extraction from different result pages. Result-Collector and Record-Collector are interfaces. Sub-classes of Record-Collector abstract class implement different record identification strategies i.e. Record-Identifier-By-Automatic-Wrapper, Record-

Identifier-By-Lixto and Record-Identifier-By-Data-Mining. Sub-classes of Result-Collector abstract class implements two result identification strategies i.e. Result-Identifier-By-Ontology and Result-Identifier-By-Lixto.

4.3. Result Merger

Result extracted from different search engines need to be merged and then stored for future use. Results can be stored in a database or in XML format. We re-used result merger design pattern for result merger component from [8] with small changes.

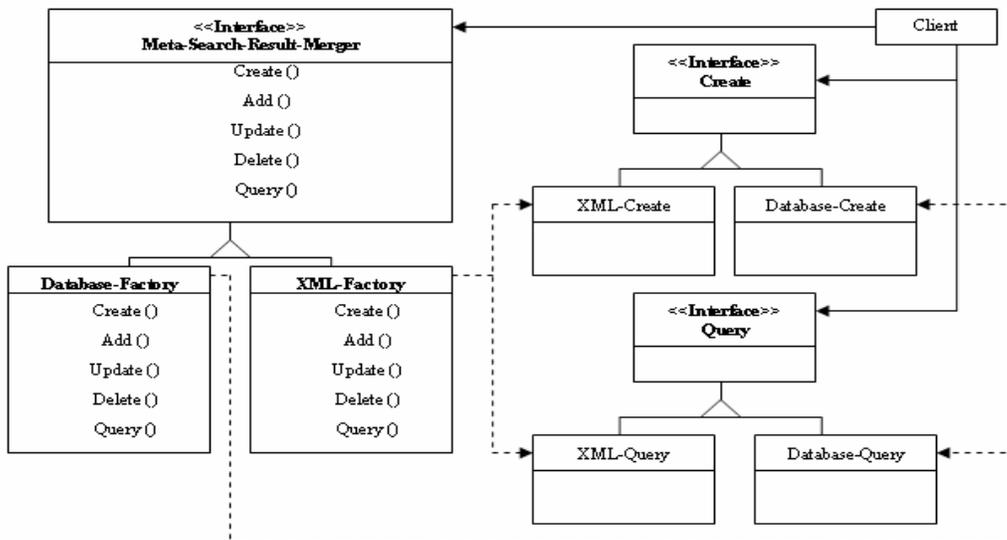


Fig. 7: Abstract Factory Pattern for Meta -Search Result Merger

The abstract factory pattern is used for the result-merger component of the meta-search engine (see Figure 7). “Meta-Search-Result-Merger” abstract factory defines the interface that all concrete factories i.e. Database-Factory and XML-Factory must implement, which consists of a set of methods for merging results. The concrete factories i.e. Database-Factory and XML-Factory implement the different product families i.e. Create, Query etc. To merge results, the client uses one of the factories and each factory knows how to create the right object for the right merging process. Few parts in Figure 7 are not drawn for the sake of saving space.

4.4. Result Ranker

A result ranker component ranks the results according to user preferences. Rank preferences can vary according to personal choices or meta-search engine type (i.e. job, hotel). For example, a seeker may want to rank the flight results according to the price, hotel according to the nearest location or job according to the query relevance. For merging results into a single ranked list according to the query relevance aone of the algorithm from TopD, TopSRR, SRRSim, SRRRank ,SRRSimMF [17] can be used.

A

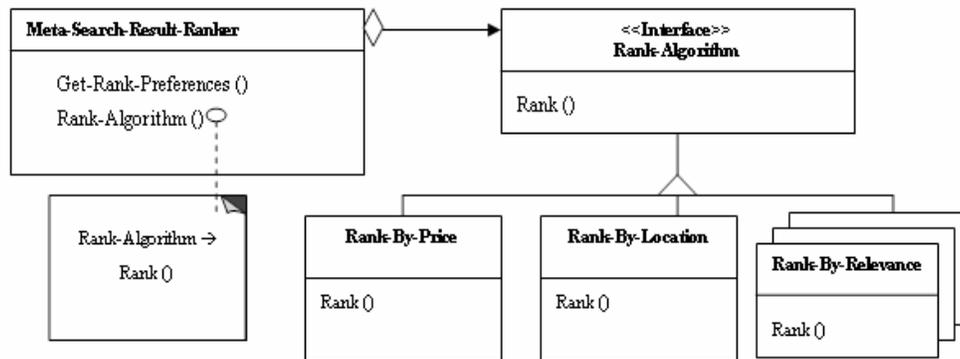


Fig. 8: Strategy Pattern for Meta-Search Result Ranker

strategy design pattern is used for the result-ranker component of the meta-search engine (see Figure 8). The Meta-Search-Result-Ranker class is responsible for ranking of results. Sub-classes of the Rank-Algorithm abstract class implement different ranking strategies i.e. Rank-By-Price, Rank-By-Location, and Rank-By-Relevance.

5. Conclusion

In this paper, we presented design patterns that can structure complex meta-search engines construction processes and provides us with flexible design strategy. We introduce design patterns for meta-search engine construction and usage as well as its components.

The difference between our and existing work is that we also introduced design patterns for i) a query interface generator component of meta-search engines that is required for user convenience and ii) information extraction component for information extraction from search engine's result pages.

The reusable design patterns for meta-search components can be reused several times not only in meta-search domain but also in some different application domain after some modifications. Design patterns for meta-search engines are flexible enough and have facility to add or remove features for different components with minimum effort in future. These design patterns can speed up the development process for new developers as they do not need to rediscover the design problems and can save time. These design patterns for meta-search also enhance communication between project team members by providing them with shared vocabulary and provide a way to alter or extend some part of the system independently of all other parts.

The extend of reuse of software and software designs is difficult to evaluate since a reuse typically occurs years later after the design process. Moreover, the critical aspect is the reuse of software by developers not involved in the original design.

We have designed the design patterns with three applications in mind (job search, accommodation search, and search for transport orders). Moreover we have analyzed different algorithms described in the literature that can be used in meta-search.

References

- [1] Lee Rainie, and Jeremy Shermak, Search Engine Use, Pew Internet and American Life tracking Project, http://www.pewinternet.org/pdfs/PIP_SearchData_1105.pdf , 2005
- [2] Bill Gates, Beyond Business Intelligence: Delivering a Comprehensive Approach to Enterprise Information Management, <http://www.microsoft.com/mscorp/execmail/2006/05-17eim.mspx>, 2006
- [3] Kim Petterson, How search is redefining the Web and our lives, The Seattle Times, http://seattletimes.nwsourc.com/html/localnews/2002259118_search01.html,2005
- [4] Tony Hammond, Timo Hannay, Ben Lund , and Joanna Scott, Social Bookmarking Tools (I), D-Lib Magazine, Volume 11, Number 4, ISSN 1082-9873, 2005
- [5] Yaffa Aharoni , Ariel J Frank, and Snunith Shoham , Finding information on the free World Wide Web: A Specialty meta-search engine for the academic community, First Monday, Volumr 10, Number 12 http://firstmonday.org/issues/issue10_12/aharoni/index.html, 2005
- [6] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995
- [7] Eric Freeman, and Elisabeth Freeman, Head First Design Patterns, O'Reilly, 2004
- [8] Junlin Zhang, Weimin Qu, Lin Du, and Yufang Sun, A Framework of Domain-Specific Search Engine: Design Pattern Perspective, IEEE International Conference on Systems, Man and Cybernetics, Volume 4, Page 3881- 3886, Washington, D.C. USA, 2003
- [9] Hai He, Weiyi Meng, Clement Yu, and Zonghuan Yu, Automatic Integration of Web Search Interfaces with WISE-Integrator, The International Journal on very large databases, VLDB, Volume 13, Number 3: 256-273, 2004
- [10] Kevin Chen-Chuan Chang, Bin He, and Zhen Zhang, Towards Large Scale Integration: Building a MetaQuerier over Databases on the Web, In Proceedings of the Second Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, California,2005
- [11] Bin He, Zhen Zhang, and Kevin Chen-Chuan Chang, Towards Building a MetaQuerier: Extracting and Matching Web Query Interfaces, Proceedings of 21st International Conference on Data Engineering, Tokyo, Japan 2005
- [12] Jürgen Dorn, and Tabbasum Naz, Integration of Job Portals By Meta-Search, Proceedings of 3rd International Conference on Interoperability for Enterprise Software and Applications, Funchal, Portugal, 2007
- [13] Jürgen Dorn, Tabbasum Naz, and Pichlmair Markus, Ontology Development for Human Resource Management, 4th International Conference on Knowledge Management, Vienna, Austria, 2007
- [14] Jürgen Dorn, Peter Hrastnik, Albert Rainer. and Peter Starzacher (2008) [Web Service based Meta-search for Accommodations](#), *Information Technique & Tourism Journal*
- [15] Hongkun Zhao, Weiyi Meng, Zonghuan Wu, Vijay Raghavan, and Clement Yu, Fully Automatic Wrapper Generation for Search Engines, Proceedings of the 14th International World Wide Web Conference, Page 66-75, Chiba, Japan, 2005
- [16] Robert Baumgartner, Sergio Flesca, and Georg Gottlob, Visual Web Information Extraction with Lixto, Proceedings of the 27th VLDB Conference, Rome, Italy, 2001
- [17] Bing Liu, Robert Grossman, and Yanhong Zhai, Mining Data Records in Web Pages, SIGKDD, Washington, DC, USA, 2003
- [18] Yiyao Lu, Weiyi Meng, Liangcai Shu, Clement Yu, and King-Lup Liu, Evaluation of Result Merging Strategies for Metasearch Engines, 6th International Conference on Web Information Systems Engineering (WISE05), Page 53-66, New York City, 2005.