aKIA Automobile: a Multi-Agent System in E-Commerce

Abbas Kia, Esma Aïmeur and Peter Kropf

University of Montreal
Department of Computer Science and Operations Research
Montreal, Quebec, Canada
{kiaseyed, aimeur, kropf}@iro.umontreal.ca

Abstract

Electronic commerce is a domain where agent technologies are well suited. The search and retrieval of product information is crucial in every E-commerce system. While some systems provide the user just with a fixed selection of products, others offer a wide variety of products with different characteristics. In this case, the system must provide means to efficiently and accurately search and retrieve product information according to the user's desired product characteristics. In this paper, a three-layer architecture specifically tailored for data retrieval in a distributed network data system is presented. In particular, it includes search methods allowing for measuring the matching rate between the user's query and the products found. These methods allow to easily re-formulate the user's queries for better matching his request. The architecture has been implemented and tested in the case of a second-hand automobile market. The practical results obtained are encouraging in that the system works as expected and shows promising performance characteristics.

Keywords E-Commerce, Agent Architectures, Retrieval Agent & Methods

1. INTRODUCTION

Electronic commerce is a popular domain for the application of agent technologies [Kristensen, 2001]. Indeed, many systems have been proposed and implemented for product information retrieval, auctioning, brokering and negotiating, etc. An important element for any E-commerce system is the efficient, accurate and precise search and selection of products a user wishes to purchase.

Agents support the buying and selling of products and services in the Internet for their users [Chavez and Maes, 1996; Brenner, 1998]. Often, the users need to reformulate their queries to accomplish effective product information retrieval, because the result is either insufficient, or too much information (or very large lists of documents) is delivered.

This paper presents a three-layer agent architecture allowing for efficient product information retrieval and selection. To demonstrate our approach, a system (called aKIA) comprising a user interface and a filtering/retrieval agent has been implemented [U1]. A third agent present in the architecture for automatically gathering product information has yet to be implemented. At present, the system has been tested with product data manually collected and fed into the system for testing purposes.

2. A three-layer Agent Architecture for Product Search and Retrieval

The aKIA system is composed of three agents, the "Interface Agent", the "Filtering/Retrieval Agent" and the "Information Agent". The first and the second agent work and interact with each other. They are responsible for getting

the user's criteria, retrieving the information and displaying the results. The third agent is responsible for finding, fetching, abstracting and classifying information from distributed sources.

2.1. aKIA Architecture

Figure 1 shows the overall structure of the proposed three-layer architecture. Each layer is presented in more detail in the following sections.

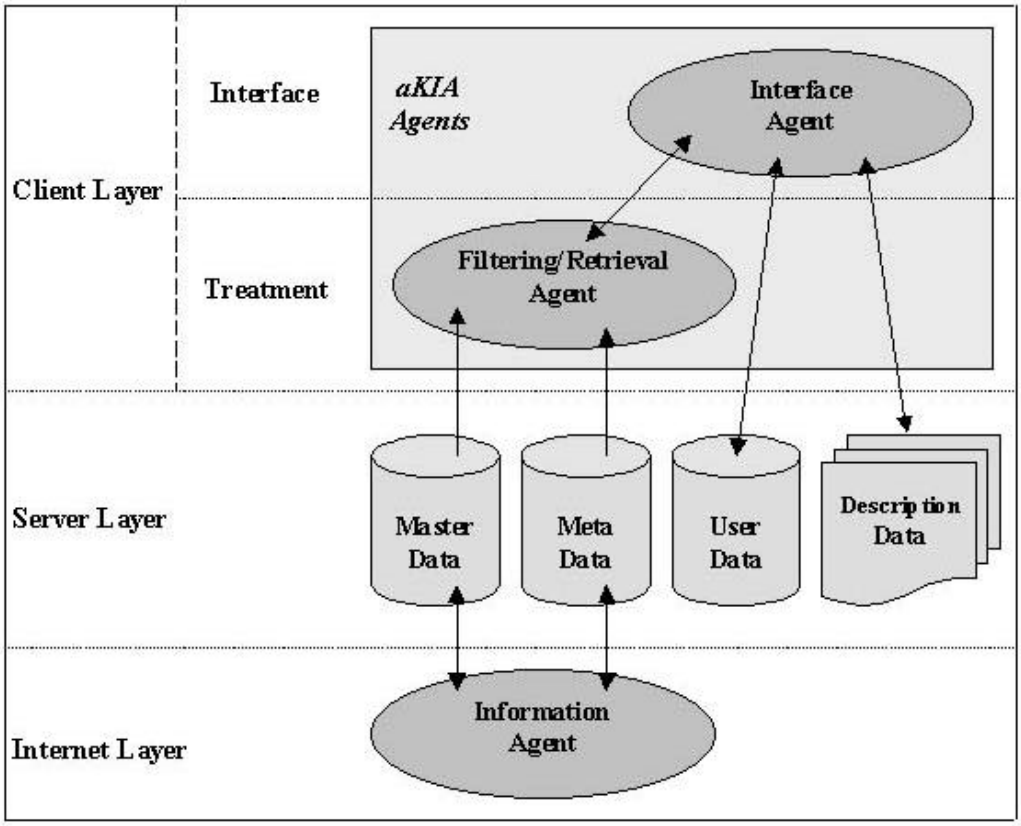


Figure 1: Three-layer agent architecture

2.1.1. Client Layer

This layer contains the agents for processing and filtering information. At this layer, the users interact with the system, and the results are displayed.

The "Interface Agent" resides in the ***Interface*** sub-layer. It provides for user's registration, gathering information from the user, and displaying information that is provided by the "Filtering/Retrieval Agent".

The "Filtering/Retrieval Agent" in the ***Treatment*** sub-layer handles the logic process. It searches the information on behalf of a user.

2.1.2. Server Layer

This layer serves as storage for data required by the system and for sharing user data. The application in the "Server Layer" is responsible for running different application modules in different processes, passing data between them and distributing them across physical processors and machines.

2.1.3. Internet Layer

The data extracted from the distributed databases by the "Information Agent" is stored in the "Master Data" where the "Filtering/Retrieval Agent" can retrieve it later.

Interactions between the client and the server operate the same way as they do in a two-layer system. The third layer provides comprehensive data services, including database operations and any other services needed to support a robust e-commerce server.

2.1.4. Interface Agent

The interface agent is a user agent that acts as an intermediary between the user and the rest of the networked world. It must be able to effectively and efficiently interact with the user. The interface, in the form of a Java applet, is accessible to the user from any Web browser whenever the user requests to interact with the application.

The "Interface Agent" provides a form to enable the users to define their search criteria, communicates with the "Filtering/Retrieval Agent", and displays results in an appropriate manner. The aKIA Interface has a direct relation with the "User Data" and the "Description Data".

2.1.5. Filtering/Retrieval Agent

This agent searches the information sources to find data that matches the selection criteria of the user. Upon receiving a request, the "Filtering/Retrieval Agent" accesses the data, looking for matching information, and returns it to the "Interface Agent". The Retrieval Methods that we have implemented will be discussed later in section 2.5.

Filtering and retrieval are seen as two sides of the same coin. Retrieval agents search and retrieve information. They are to recommend the user a ranked list of information. Filtering agents model and monitor the interests of the user. They are used to reduce information overload by removing undesired data [Stenmark, 1998] from a stream of incoming information. Information filtering recognizes the interest of a user and delivers thus content that is based on his particular interests.

The agent runs on the user's workstation at all times. A mobile agent could be used instead to move between data sources and check the data locally. It is vital that the requested information satisfies the user's query and goals. The request sent by the "Filtering/Retrieval Agent" must thus be based on both, the information obtained directly from the user and also upon any user model the system has managed to build up.

2.1.6. Information Agent

There is an ever-increasing amount of information available on the Web. This agent performs the role of managing, manipulating or collating information from many distributed sources and will be responsible for finding, fetching, abstracting and classifying the actual information that the user is interested in. Information agents are tools to help to manage the tremendous amount of information available.

The "Information Agent" requests particular pieces of information from different information sources and allows the user to deal with distributed sources of information effectively. As stated before, this agent still needs to be implemented.

2.1.7. Databases

We have four types of data including: *Master Data*, *Meta Data*, *User Data* and *Description Data*.

Master Data: This is a database, in which the "Information Agent" stores the product data extracted from many distributed sources (See Section 2.2.). The "Master Data" must contain complete and up-to-date information.

Meta Data: Meta data consist of information about the data. i.e. data characterizing the data present in the "Master Data".

User Data: All information related to the user, like registration data and their search criteria, exist in this database. During operation all client-specific parameters are tracked and stored in User Data, such that the system can be restored to its last known operating state.

Description Data: This collection holds data or files of all product descriptions, which are available to the system. Each product description gets its unique id.

2.2. The application domain

Our architecture aims to be *generic*. It is therefore possible to apply this architecture in different domains in E-commerce. To demonstrate the viability of our approach, we focus in this paper on a product with common characteristics and rather small sets of records.

The *automobile* domain was chosen as the context for this work. It is a product with common characteristics such as mark, model, etc. The accessible cases of product descriptions in the search space are small and finding them needs more time and energy than other products like books and CDs.

Our database (*Master Data*) currently includes about 350 records of automobiles gathered from the site [U2]. They are described by the following attributes: mark, model, color, transmission, fabric, kilometer, price, cylinder and door. The last two attributes are not considered in our search criteria. All the records are valid and for each record, there also exists a "Description Data".

2.3. An Example

Consider the following parameters to search for a particular vehicle type:

Mark: Mercedes-Benz

Model: CLK, S or SL

Color: black

Transmission: Automatic

Fabric: between 1995 and 2002

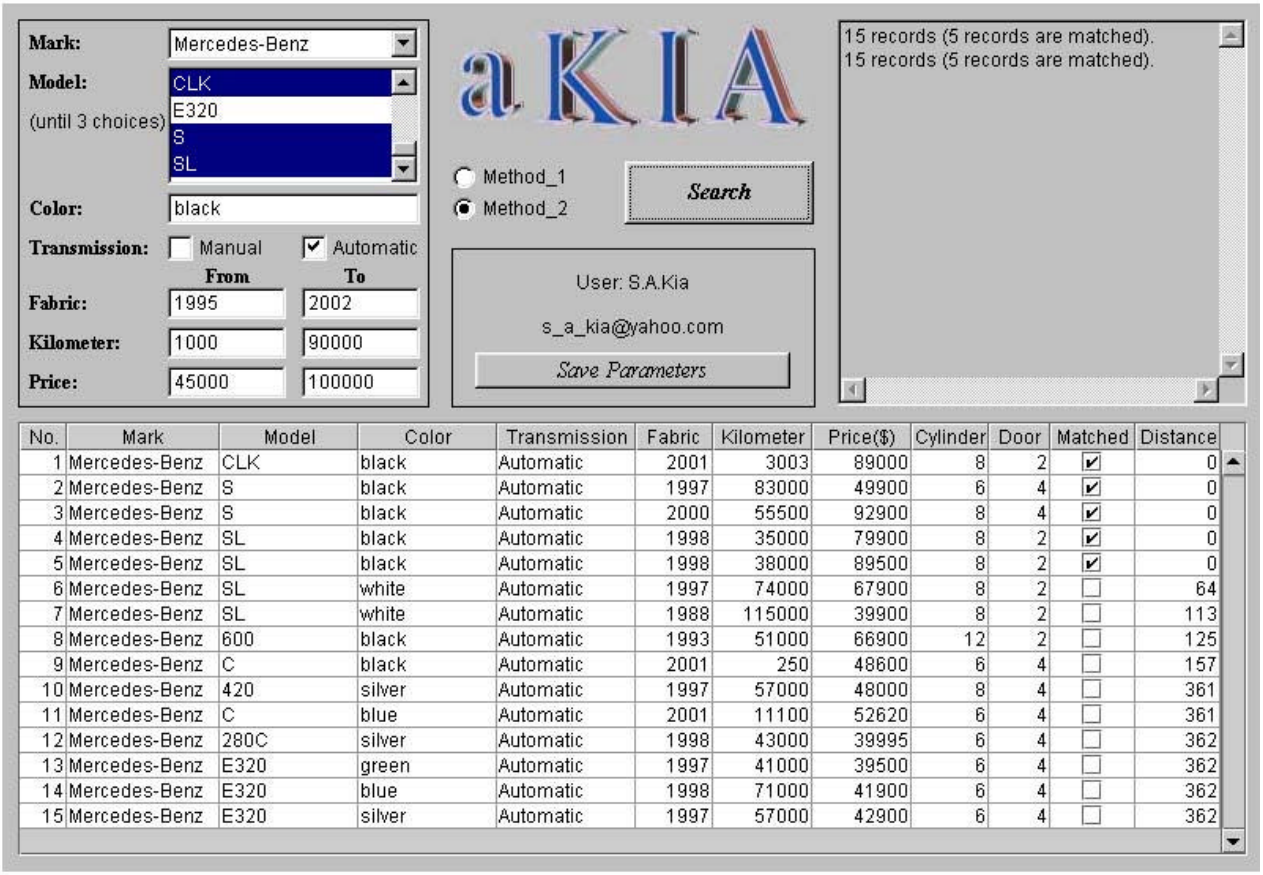
Kilometer: between 1000 and 90000

Price: between 45000 and 100000

The user enters the values of parameters and selects the search method. After that, the user just clicks the "Search" button and let the "Filtering/Retrieval Agent" do the search. The "Filtering/Retrieval Agent" will then provide the recommended results according to these parameters.

Figure 2 shows the input case and the fifteen records the "Filtering/Retrieval Agent" has found. Only the first five records are matched.

By clicking each row of the result table, the user can view the *Description Data* of that case in order to consult the complete product description.



No.	Mark	Model	Color	Transmission	Fabric	Kilometer	Price($)	Cylinder	Door	Matched	Distance
1	Mercedes-Benz	CLK	black	Automatic	2001	3003	89000	8	2	☑	0
2	Mercedes-Benz	S	black	Automatic	1997	83000	49900	6	4	☑	0
3	Mercedes-Benz	S	black	Automatic	2000	55500	92900	8	4	☑	0
4	Mercedes-Benz	SL	black	Automatic	1998	35000	79900	8	2	☑	0
5	Mercedes-Benz	SL	black	Automatic	1998	38000	89500	8	2	☑	0
6	Mercedes-Benz	SL	white	Automatic	1997	74000	67900	8	2	☐	64
7	Mercedes-Benz	SL	white	Automatic	1988	115000	39900	8	2	☐	113
8	Mercedes-Benz	600	black	Automatic	1993	51000	66900	12	2	☐	125
9	Mercedes-Benz	C	black	Automatic	2001	250	48600	6	4	☐	157
10	Mercedes-Benz	420	silver	Automatic	1997	57000	48000	8	4	☐	361
11	Mercedes-Benz	C	blue	Automatic	2001	11100	52620	6	4	☐	361
12	Mercedes-Benz	280C	silver	Automatic	1998	43000	39995	6	4	☐	362
13	Mercedes-Benz	E320	green	Automatic	1997	41000	39500	6	4	☐	362
14	Mercedes-Benz	E320	blue	Automatic	1998	71000	41900	6	4	☐	362
15	Mercedes-Benz	E320	silver	Automatic	1997	57000	42900	6	4	☐	362

Figure 2: aKIA Interface

2.4. Distance Function

A distance function has been introduced to evaluate the users' degree of satisfaction. The smaller the distance, the better the results will match the desired parameter values.

The product attributes in our database are divided in two categories. The first category contains the attributes with an exact and precise value. The values of the attributes of the second category are of interval type. The attributes of the first category are: mark, model, color and transmission. The attributes of the second category are: fabric, the construction year (fabric ? $\{f_1, \ldots, f_2\}$), kilometer, the tachometer stand (kilometer ? $[k_1 \ldots k_2]$), and price, the amount asked for the product (price ? $[p_1 \ldots p_2]$).

For measuring the distance between each record and the user's query, we use the following function:

$$distance = (\Sigma_{f=\{\text{mark, model, color, transmission}\}} d_f)^2 + (\max_{s=\{\text{fabric, kilometer, price}\}} d_s)^2$$

This distance function assigns to each record a unique non-negative integer value.

d_f: distance of the attributes of the first category from the user's criteria.

d_s: distance of the values of the attributes of the second category from the user's criteria.

This can be reformulated as:

$$distance = (d_{\text{mark}} + d_{\text{model}} + d_{\text{color}} + d_{\text{transmission}})^2 + (\max(d_{\text{fabric}}, d_{\text{kilometer}}, d_{\text{price}})^2$$

We consider the maximum distance when calculating d_s rather than the sum of them, because the maximum covers the others. There is the value n for controlling the highest and the lowest limit of the distances. We take into account this value when calculating d_s for the maximum acceptable distance of fabric. The highest kilometer and price should not be n times higher than the query specifies.

The distances of each attribute are sketched in Tables 1 to 4. In Table 1, as an example, a value of 8 for the distance of model means that an automobile with a model different from the one requested has a distance equal to an automobile with the fabric with 8 years difference from requested. Other distances for these attributes do not disturb our function. But it is necessary to use greater distance for the attributes that are more important.

distance	**equal with request**	**different from request**
d_{mark} ? {0, ? }	0	?
d_{model} ? {0, 8}	0	8
d_{color} ? {0, 11}	0	11
$d_{transmission}$? {0, 14}	0	14

Table 1: the distance of mark, model, color and transmission from the user's criteria (d_f)

< f_1 – n	**< f_1**	**in interval**	**> f_2**	**> f_2 + n**
?	f_1 – fabric	0	fabric - f_2	?

Table 2: distance of fabric (d_{fabric})

< k_1	**in inte rval**	**> k_2**	**> (n + 1) * k_2**
n - ?(n * kilometer / k_1)?	0	?(10 * (kilometer - k_2) / k_2)?	?

Table 3: distance of kilometer ($d_{kilometer}$)

< p_1	**in inte rval**	**> p_2**	**> (n + 1) * p_2**
n - ?(n * price / p_1)?	0	?(10 * (price - p_2) / p_2)?	?

Table 4: distance of price (d_{price})

2.5. Retrieval Methods

We have implemented two different retrieval methods.

In the first method (section 2.5.1), we use the distance function of the previous section for the retrieval. The distance of attributes in this method is like the weight of attributes in the classical *nearest neighbors* method that are used in many *Case-Based Reasoning* recommendation systems [Aïmeur and Vezeau, 2000].

The second method (section 2.5.2) is based on a loop, which increases the intervals and searches the database until the number of records found is equal to a constant c. This alternative new method is qualitatively comparable with the first method (see section 3 for a performance evaluation).

2.5.1. Direct Method

The first retrieval method, called *direct method*, is sketched as follows:

- For each record in the database
  - Evaluate the distance of the record from the user's query;
  - If the distance is not infinite, then add it to the temporary array;
- Sort the temporary array by distance in ascending order;
- Put the first c records or more (with the same distance than the c^{th} record) in the result array.

2.5.2. Shortcut Method

The second retrieval method, called *shortcut method*, is defined as follows:

- For **i**=0 to m
  - Change the set of attributes of the first category to be considered (effective in d_f);
  - For **j**=0 to n
    - Increase the interval of the attributes of the second category (effective in d_s);
    - Search the database for matching results;
    - If the results are not in the result array, then add them to it;
    - If the number of elements in the result array ? c, then exit.

Table 5 shows the consideration of the attributes of the first category in each step of this method. According to this Table, the value m in this method is equal to 7.

i	**Mark**	**Model**	**Color**	**Trans.**	? $\mathbf{d_f}$
0	☞	☞	☞	☞	0
1	☞	☞	-	☞	8
2	☞	-	☞	☞	11
3	☞	☞	☞	-	14
4	☞	-	-	☞	19
5	☞	☞	-	-	22
6	☞	-	☞	-	25
7	☞	-	-	-	33

Table 5: Consideration of the attributes of the first category in each step of shortcut method

Table 6 shows the formula for increasing the interval of attributes of the second category. The value n in this Table is the same value that we used in Tables 2 to 4.

f_1	f_2	k_1	k_2	p_1	p_2
$f_1 - j$	$f_2 + j$	$k_1 - k_1 * j / n$	$k_2 + k_2 * j / (m + 1 - i)$	$p_1 - p_1 * j / n$	$p_2 + p_2 * j / (m + 1 - i)$

Table 6: Formula of increasing the interval of attributes of the second category

3. Performance Evaluation

The aKIA system has been implemented using the Java programming language. Thus, it will run on any platform with a Java Virtual Machine [Bigus, 1998]. Java implements a number of security features [Federrath, Krone, and Schoch, 2001] and provides the advantages of architecture neutrality and portability to agent developers [Michaud, 1998]. aKIA allows users with no background in intelligent agent technologies to quickly and easily work with the system.

For evaluating the system performance, we executed the two methods many times (10,000 times) automatically.

The performance evaluation procedure is sketched as follows:

? For loop=1 to 10000

¦ Generate parameters randomly for each attribute;

¦ For select=1 to 2

? Execute method(select);

? Calculate time of execution;

? Calculate sum of the distance in the result table;

The time of execution of the direct method depends on the sorting algorithm that we use. We have implemented two different sorting algorithms. The first is a merge sort with a O(NlogN) performance, and the second is a selection sort with a $O(N^2)$ performance. Both, merge sort and selection sort are stable sorting algorithms, meaning that two elements that are equally ranked in the array will not have their relative positions flipped. The results for some databases with different number of records are summarized in Tables 7 and 8.

# of records	Time(1)	Time(2)	Time(2)/Time(1)	Distance(1)	Distance(2)	Distance(2)/Distance(1)
500	4.5 s	5.0 s	**111%**	32789000	33845000	**103%**
1000	10.0 s	8.4 s	**84%**	25256000	26142000	**103%**
5000	72 s	59 s	**82%**	12204000	12641000	**103%**
10000	147 s	123 s	**83%**	8985000	9282000	**103%**

Table 7: Performance evaluation with merge sort

# of records	Time(1)	Time(2)	Time(2)/Time(1)	Distance(1)	Distance(2)	Distance(2)/Distance(1)
500	12.5 s	5.0 s	**40%**	32221000	33318000	**103%**
1000	42.0 s	8.4 s	**20%**	25733000	26673000	**103%**
5000	920 s	59 s	**6%**	12109000	12528000	**103%**
10000	3620 s	123 s	**3%**	9003000	9302000	**103%**

Table 8: Performance evaluation with selection sort

On average, we believe that the performance of the shortcut method is acceptable (with a 3% error rate), and it is faster than the direct method. Its implementation is easy and we do not need to use sorting algorithms.

A comparison of the performance is illustrated in Figure 3.

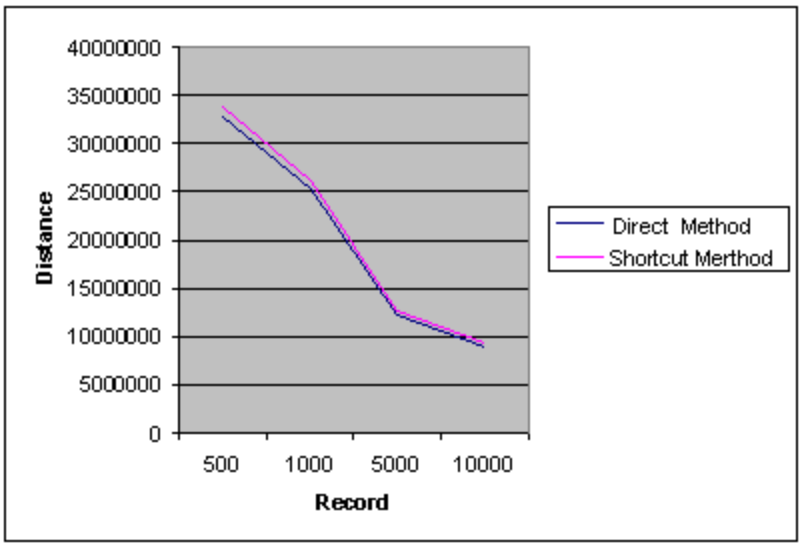


Figure 3: Comparison of the performance

It is evident that when the number of records increases, the probability of finding the matching records also increases. Consequently, the sum of distance values decreases. It is also evident that with the increasing of number of records, the direct method using selection sort is completely inefficient.

A comparison of the execution time is illustrated in Figure 4 with a logarithmic scale.

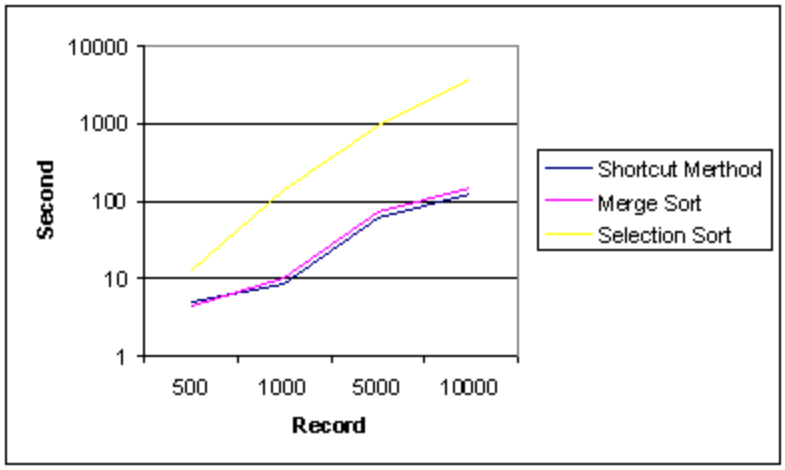


Figure 4: Comparison of the execution time

4. Conclusion and Future Work

The architecture presented here is very general and intends to be generic. The implementation example concentrates only on a suitable, domain-specific retrieval of product information in a rather standard Web/Java environment. This was believed to be beneficial for the further work now being undertaken. The tests realized with the two retrieval methods confirm the suitability of the use of case-based reasoning method in order to provide the user with more precise responses to product information requests. The performance analysis shows that the additional costs induced by such methods are acceptable.

While the implementation presented just addresses questions related to product information retrieval and presentation, work has already started to implement an instance of the generic agent architecture. This further going study will include the implementation of all the agents and components of the architecture applying various agent platforms (such as Jack and FIPA based platforms) as well as a Java (RMI, Javaspaces, …) 'from scratch' implementation. It is thereby hoped to get a better insight of the feasibility, flexibility, robustness, performance, maintainability and adaptability of the system. Furthermore, it is also planned to apply those prototypes to other application domains.

Acknowledgement

We would like to thank Dr. Seyed-Jalal Kia for his valuable comments and help.

References

[Aïmeur and Vezeau, 2000] Aïmeur, E., and Vezeau, E., "Short-Term Profiling for a Case-Base Reasoning", Proceedings of the 11th European Conference on Machine Learning, Barcelona, Catalonia, Spain, p. 23-30, May/June 2000.

[Bigus, 1998] Bigus, J., "Constructing intelligent agents with Java: a programmer's guide to smarter applications", Wiley & Sons, Inc., United States, 1998.

[Brenner, 1998] Brenner, W., "Intelligent software agents: foundations and applications", Springer, Germany, 1998.

[Chavez and Maes, 1996] Chavez, A., Maes, P., "Kasbah: An Agent Marketplace for Buying and Selling Goods", Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology (PAAM '96), London, p. 75-90, 1996.

[Federrath, Krone, and Schoch, 2001] Federrath, H., Krone, O., and Schoch, T., "Making Jini Secure", the Fourth International Conference on Electronic Commerce Research (ICECR-4), p. 276-286, Nov. 2001.

[Fischmeister, 2000] Fischmeister, S., "Building Secure Mobile Agents: The Supervisor-Worker Framework", Diploma Thesis, Technical University of Vienna, p. 6-12, Feb. 2000.

[Kristensen, 2001] Kristensen, T., "Software Agents in a Collaboration Learning Environment", International Conference on Engineering Education (ICEE), Session: Distance Learning 2, Norway, Aug. 2001.

[Michaud, 1998] Michaud, L., "Intelligent Agents For Network Management", Computer Science Department Institute of computer Communications and Applications, p. 8-10, June 1998.

[Stenmark, 1998] Stenmark, D., "Agent Classification", 1998. http://w3.informatik.gu.se/~dixi/agent/class.htm

[U1] http://www.iro.umontreal.ca/~kiaseyed/aKIA.html

[U2] http://www.fr.megawheels.com/