



Retrieving Indexed Information Application (RIIA)

Author: Fabricio Rusu-Banu

Department of Electrical and Computer Engineering
University of Calgary
frusuban@ucalgary.ca

Abstract: *This paper present a brief description of a MAS application named Retrieving Indexed Information Application (RIIA) to retrieve automatically via email or web page delivery personalized information from Internet or Intranet. The agents are namely designed to select the new documents founded in accordance with the user's needs via a filtering and indexing process. The idea is not a new one for sure, there are large or small companies dealing with this kind of product and "information management", but their products are expensive and their documentation is available only for internal purpose...*

The purpose of this application is to allow whatever user to spend less time looking and sorting information and to let the computer to do alone this extremely time consuming job.

Key words: *Multi-agent System (MAS), Agents, Agent-based Systems and Designs, Retrieving Indexed Information Application (RIIA)*

TABLE OF CONTENTS

1. Introduction.....	3
2. General consideration.....	3
3. The RIIA features.....	4
3.1. Customized information.....	4
3.2. Searching an viewing the documents.....	5
3.3. Classification of the documents.....	5
3.4. Administration.....	5
4. The RIIA Architecture.....	6
4.1. The RIIA data handling and repository.....	6
4.2. The RIIA Profiler.....	7
4.3. Agent Control Panel and Delivery Control Panel.....	8
4.3.1. Example of using libRIIAstd library.....	9
4.3.2. Example of using libRIIASmtp library.....	10
4.4. User database.....	10
5. RIIA agents delivering information.....	11
5.1. User information.....	12
5.2. Defining agents.....	13
6. Conclusion.....	15
Bibliography.....	14

1. Introduction

The general purpose of *Retrieving Indexed Information Application* (RIIA) is to let the computer to do this job alone using its agents. The idea came up mostly inspired by the frustrating Canadian job hunting. I'm perfectly convicted that this topic has been debated on this course or whatever over and over again. But I must confess the practical idea has been developed in my mind after spending ours and ours on the career web sites. I was asking myself "*Why I would not let the computer doing this kind of job alone?*", not necessarily focused on job-hunting activities, but searching, analyzing, retrieving and indexing all kind of documents from the Internet or whatever Intranet.... Moreover it would be great to receive some messages, often emails letting me know about new postings in the web sites already scanned in the previous days. Maybe something was posted over there in the meantime... The information retrieved would be accordingly with some indexes because I'd like to receive information accordingly with my personal key indexes, not overall information, useless to my personal need.

The *Retrieving Indexed Information Application* (RIIA) can be used in information management activities, where is necessary to automatically search, analyze through some indexes or filters a huge amount of documents posted whatever on Internet or Intranet, sorting and delivering the results of accordingly with dedicated area of interest, such engineering, financing, media, healthcare, whatever is necessary to deal with a huge amount of documents especially when it's necessary to obtain a particular document containing a particular set of information. But for large scale and more evaluated purposes, it is necessary server version, which the administrative level can assign users, specific agents in accordance with company or department specific field of interest, user/group management, *Agent Control Panel* and *Delivery Control Panel Management*. I will schematize only briefly about the higher version and capabilities of the RIIA application.

For sure there are some companies selling out on the market this kind of products, but generally this application are extremely expensive.

2. General consideration

An agent is an object that searches, watches, retrieve, deal for information based on a user criteria. Agents include the agent name, the identification of the agent's owner, a query, then a specified action accompanied by a set of arguments that more specifically describe the action [[Far, 2004](#) and [AgentBuider RM, 2000](#)]. This following table explains a defined agent and the data it might contain:

Name of the Agent	Job postings
Owner	Fabricio Rusu-Banu
Query	Software engineer, system engineer, mechanical engineer, operations manager

Action	Invoke system action
Arguments	Send a email to the owner containing the information retrieved with a document title in the subject line

The user, in this case the user Fabricio Rusu-Banu has activated an agent, the system searches for any documents that contain words “*software engineer*”, “*system engineer*”, “*mechanical engineer*”, “*operations manager*”, etc. When it finds such a document it emails it to the agent’s owner. The agents created (could be so many, each string of research can be a distinctive agent) are stored in a repository, loaded or unloaded by an “observer” application. The agents can be stored in a database or developers can create a custom agent.

3. The RIIA features

The RIIA has two versions: first one, a simpler version, is a user application for the only one user/customer. It only requires a web server installed, and a SMTP web server (often it comes with the same package with the web server) or Microsoft Exchange if he decides to receive the automatically updated information by emails. The second one is more complex, a client-server application, where an Administrator has the right to administrate, configure, assign the rules/rights in accordance with the company/department area of interest, or kind of information used by these groups/users. I will explain only short hints about the client-server version, pointing out few ideas or some hints for a possible work.

The RIIA feature allows monitoring dynamically the sources of information and retrieving such type of information by relevant and timely information accordingly with the user’s needs. The server operating systems must have installed Web server (IIS, Netscape Server, Apache, Lotus), and SMTP server applications, like Microsoft Exchange.

The RIIA purpose is to retrieve documents from Internet, Intranet through a RIIA profiler accordingly with the user’s needs and area of interests. The profiler would select documents following the user’s indexes and filtering configurations. Then automatically through a number of agents, the user will receive notifications about new documents founded whatever on the Internet or Intranet by email, new web page delivery, etc.

3.1. Customized information

The RIIA agents alert the users when new relevant information has been posted on the web pages like new pages, or added information on the same web HTML page. The user basically will be able to configure their personal agents to monitor continuously the indexed web addresses. When the agent finds new information posted whatever in Internet or Intranet, the user receives an

automatic notification, obviously through an *email*. But the delivery can be as well be a *system action* (system call), *news page delivery* (each time the user accesses the news page, the RIIA collection is queried to obtain all transactions), *write action* (the agent will write a file in the user's system), *email to pager*, or send a *message to SMS*. The RIIA can to be customizable by programming the user's own delivery methods, too.

The Email delivery method provides information to user's email address. The Web method is to stores the information found on the Internet/Intranet on a repository and delivers them to users as hypertext links on a News Page Delivery. The pager or SMS use the email gateway set up to handle delivering email to an alphanumeric pager or SMS mobile phone services.

3.2. Searching and viewing the documents

The type of information to be received can be particularized through an indexing process, the user can define filters. Then the agents can deliver the information, and then the users can navigate the content of the information.

The agents deliver information to users. The agent's delivery action

3.3. Classification of the documents

The users can categorize/sub-categorize accordingly with his personal area of interest using filters. The new information is added to the categories/sub-categories automatically when is available.

3.4. Administration

The RIIA is administered through an HTML interface (it's the simplest way, the API is robust and simple, plus I don't think it would be necessary a particular GUI application to be developed for namely this purpose). The customized options can be customized through APIs. The user's view of personal agents can be set-up in HTML pages using searching scripts.

3.5. The communication protocol

The communication protocol is HTTP as low-level. Java and C/C++ are good implementation languages because these are built up on networking functions, providing XML parsing API's recently.

4. The RIIA architecture

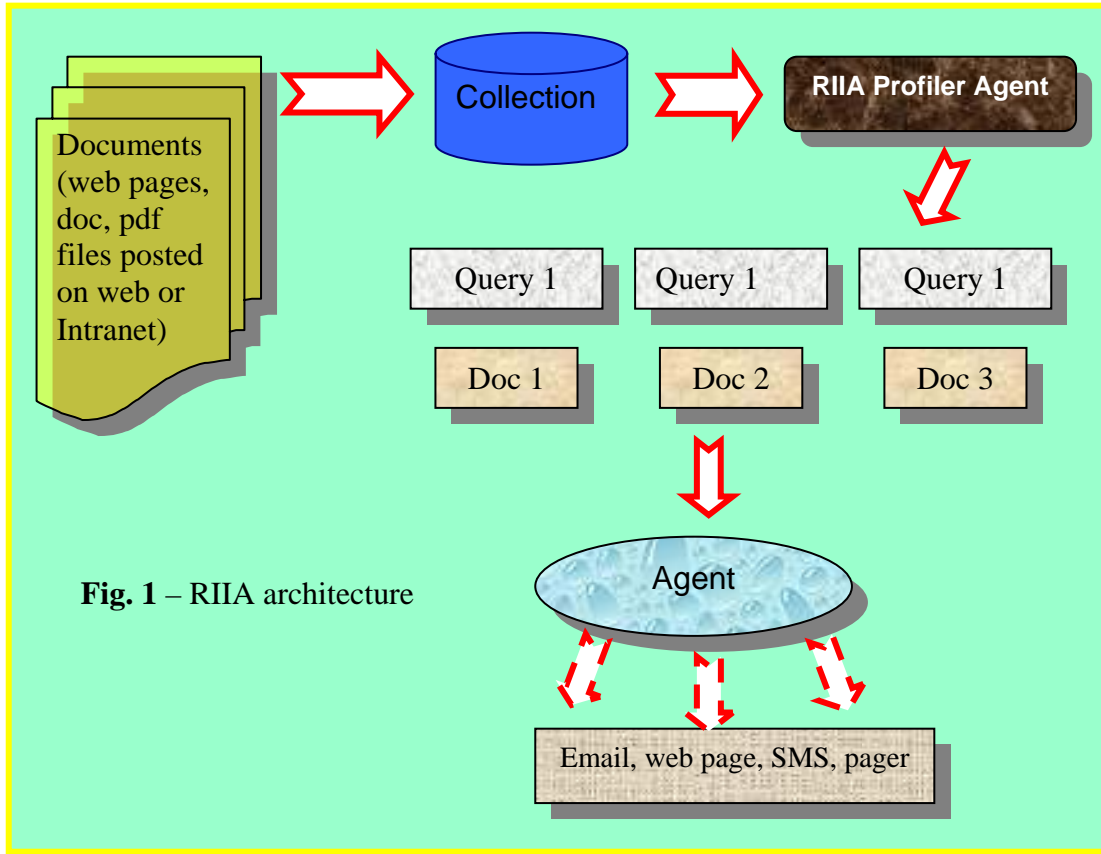


Fig. 1 – RIIA architecture

The principle sounds like that:

- ◆ New documents (html, txt, doc, pdf) from Internet or Intranet enter into collection accordingly with the user's criteria
- ◆ The RIIA profiler monitor and examines in real-time the collection, notices the new documents by comparing them with those existing already in the existing agent queries
- ◆ When the RIIA profiler identifies a new document that matches the agent query, it sends an email or system notification to the user

4.1. The RIIA data handling and repository

The agent definitions are stored into a *repository*. The documents meeting the agent search criteria can be delivered to a repository. The RIIA subsets fit with an RDMS, or an ODBC-compliant database. But basically a user will be able to define its particular repository in a simpler format.

The RIIA application data suite allows to the way a specific data is managed. These applications consist is several functions and I will try to express them in way as much as appropriate from a correct coding one:

```
class RIIADataNew {
/* this class creates a new application data handle */
int RIIARespository(repository);
char RIIAData(*appData);
char RIIADataNewArgRec* (newData);    }

class RIIADataFree {
/* this class frees RIIA Data */
void RIIAData ();    }
```

The RIIAAppData is returned by RIIADataNew

```
class RIIADataGetInfo {
/* this class provides information about the application data */
char RIIAData (*Data);
char RIIADataGetArgRec* (getArg);
char RIIADataGetOut (*getOut);    }

class RIIARespositoryOpen {
/* this class opens an agent repository and initialize functions */
void session ();
int RIIARespository (repository);
char RIIARespositoryOpenArgRec * (repositoryOpen);    }

class RIIARespositoryClose {
/* this class closes an agent repository and free the memory*/
int RIIARespository (repository);    }

class RIIARespositoryGetInfo {
/* this class provides information about an agent repository*/
int RIIARespository(repository);
char RIIARespositoryGetArcRec* (getArg);
char RIIARespositoryGetOut (*getOut)    }

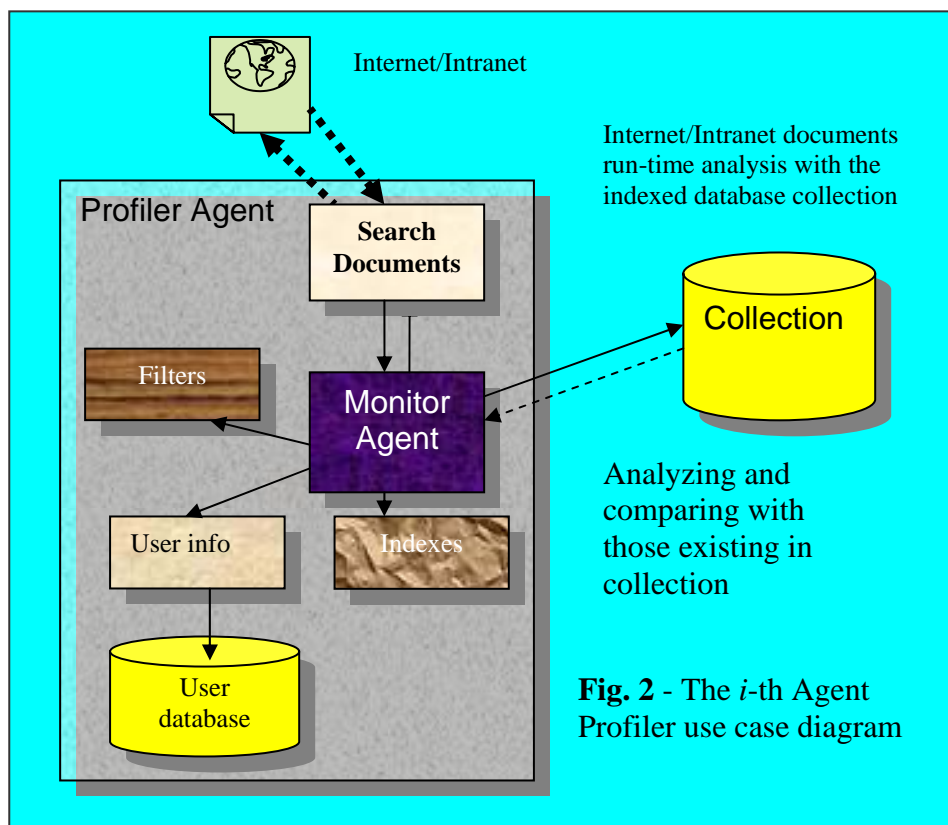
class RIIARespositorySubmit {
/* this class maintains information about an agent in repository */
int RIIARespository (repository);
char RIIARespositorySubmitArgRec* (submitArg);    }
```

4.2. RIIA profiler

The profiler is an agent and it acts like a real-time guardian. It monitors the collection(s) and activates agents when new documents accordingly with the agent queries are added. The profiler takes the new documents one by one and compares them against all the agent queries. The RIIA profiler processes one document per time. When the document matches the agent's query, the profiler

notifies the agent. Then the agent “fires” the user with an email. The profiler compares the next document against all agent queries. Finally, when the RIIA profiler has processed all the new documents, it records the state of the collection and waits for new documents to arrive.

The profiler is configured by a “RIIA Control Panel Profiler” to stop, restart the profiler, a process necessary for a new collection. It is logical to enable the profiler immediately after declaring a new collection. The “RIIA control panel profiler” also can configure several system options like *Agent Filtering* (a limitation query for particular agents, like making some properties like enabling those agents only owned by the user “Fabricio” and nobody else), *Performances*, *Log files*, *dumb files* posting, etc.



4.3. Agent Control Panel and Delivery Control Panel

The *Agent Control Panel* allows viewing the agents in repository and setting the options for the agent and the delivery subsets. It can show the agent lists, statistics, customize the agent subset and specify/re-allocate the system resources for a particular agent or set of agents. It can customize an agent subset too by assigning a path for the repository, style, type of locale used, background, etc.

The *Delivery Control Panel* allows customizing the supplied delivery subsets. It can add or delete a delivery subsets, or customize these for particular agents. As I've said briefly before, it can use a Web driver to deliver html document so they can be displayed on a user's *News Page*. The *SMTP Email* can be used for assigning the email or pager delivery methods. The principle is to assign a value for `smtp_hostname`, substituting the personal email server for the default local host. As well it can be used MAPI Email subset for emails and pagers delivery methods. This protocol is especially necessary under the Windows-based applications. The SMTP and MAPI cannot be used both at the same time. The *Delivery Control Panel* can customize the *Delivery Performance* (server resources allocated, the time that a document can be stored before to be deleted).

The delivery drivers and delivery methods are customized by the user through set of panels. When Microsoft Exchange Server is used, the username of the account under account should be the same as the Mailbox name accessed by that user's profile. Therefore the name of the mailbox on the Exchange Server should be the same with the log on to the Windows XP workstation.

For instance for these driver subsets, it should be created some delivery libraries, with specific missions:

Delivery subset	Action
libRIIStd	write, system
libdbRIIA	database
libRIIASmtp	SMTP, email, pager, SMS
libRIIAdcmc	cmc, email

4.3.1. Example of using libRIIStd library

As I've said in the previous table, libRIIStd manages the system and write actions. The write occurs in a text file in the file system. When the value of action in the agent definition is write, the system interprets actionargs as:

actionargs: "*filename args*"

After several document hits the file might appear like that:

text/doc0001	whatever@whatever.com	Workopolis
text/doc0002	whoknows@whoknows.com	IBM careers
text/doc0003	someone@someone.com	Jobs in USA

The system action executes a system call using the `system ()` library call from the standard C library. The delivery subset wraps every `${fieldname}` construct single quotes to accomplish Bourne "shell escaping". This type of escaping is compatible with the Perl `shellwords ()` function at the Bourne shell. When the value of action in the agent definition is system, the driver interprets actionargs as follows:

actionargs: "command"

4.3.2. Example of using libRIAsmtp library

The libRIAsmtp library is used to allow using the smtp, pager or SMS protocol for delivery when a new document is founded. When the value action in the agent definition is smtp or email, the subset interprets actionargs:

actionargs: 'fields'

For example the action and actionargs fields of the agent definition might look like one of the following:

```
action: "smtp"
actionargs: "TO=frusuban@ucalgary.ca, SUBJECT = ${AGENT_NAME}
agent hit"
```

```
action: "smtp"
actionargs: "TO=Fabricio, SUBJECT=new jobs posted on IBM Canada, Calgary area,
BODY=New job posting meeting your criteria has found in Calgary area
Ar:${_NL}${$}"
```

4.4. User Database

The users can be recorded or searched like in the following example:

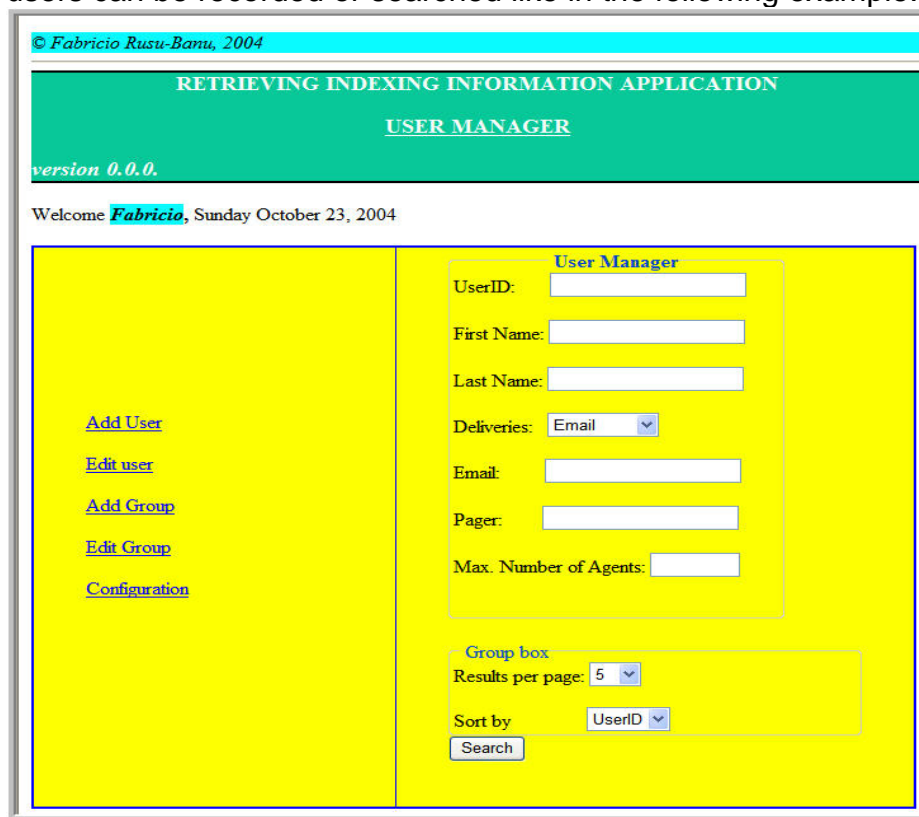


Fig. 3 – User Manager Interface

The user records that match a respective search criteria are displayed something like that:

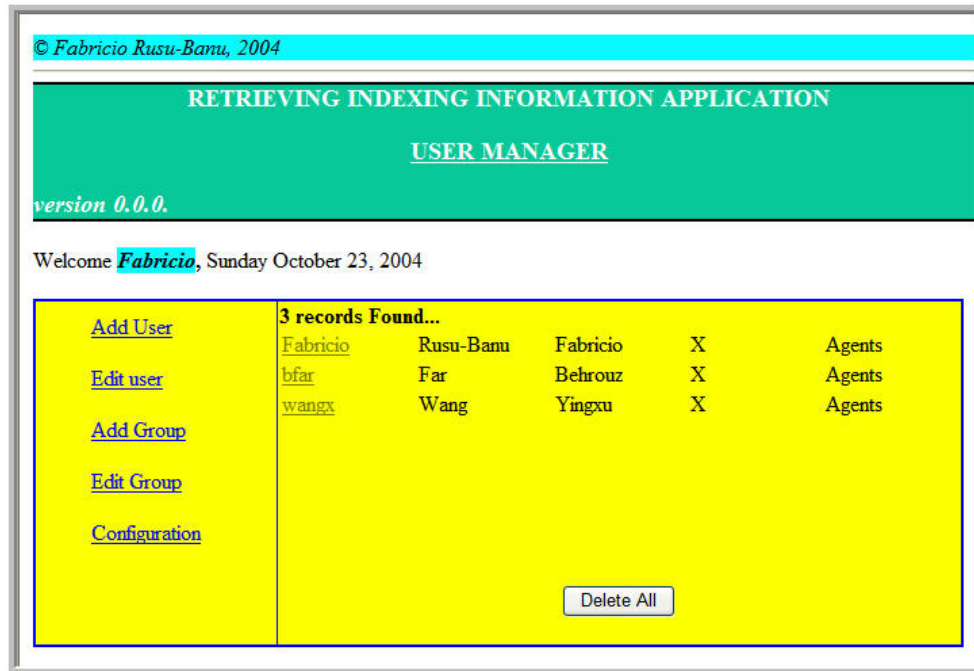


Fig. 4 – User manager database interface (retrieved users)

The UserID is a unique identifier for a user. The same is possible with the groups.

5. RIIA agents delivering information

As I've said earlier, the RIIA agents are used to deliver fresh information, whatever they are posted in whatever web server. The delivery action informs the user when a new match is found. The user can configure the agent to update a personalized web page and/or deliver an email message. An example of personalized web page is shown below:

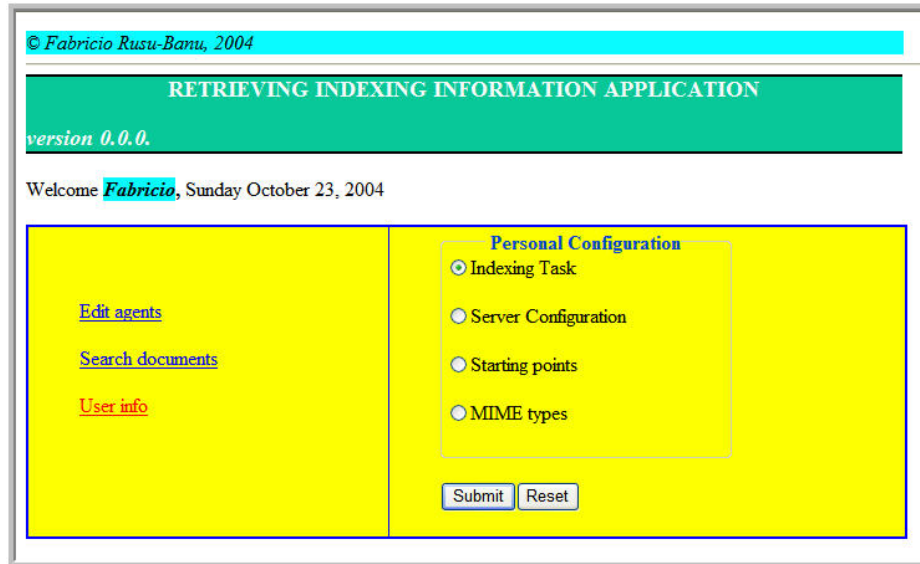


Fig. 5 – Main interface

Each user could create an agent on as a needed basis and a personal web page. Administrators can limit the number of agents a user can create accordingly the needs, the system and, who knows, personal job task, as well as filter the queries users to agents. The administrators basically would have the rights to measure and control the users' rights. These delivering informations are constructed on the API suites, some relevant examples are shown below:

```
class RIADeliverySessionNew {
/* this creates a new delivery subset session*/
int Session;
char RIADeliveySession * (deliverySession);
char RIADeliveryNewArgRec* (deliverSessionNew);    }
```

```
class RIASessionFree {
/* it frees a RIADeliverySession handle and all its associated resources for a
session handle created by a RIADeliverySessionNew call */
int RIADeliverySession (deliverySession); }
```

```
class RIADeliverySessionGetInfo {
/* this class provides information about delivery session */
int RIADeliverySession (DeliverySession);
char RIADeliverySessionGetArgRec* (getArg);
char RIADeliverySessionGetOut* (getOut);}
```

5.1. User information

The user can introduce the information about him. The display example is shown below:

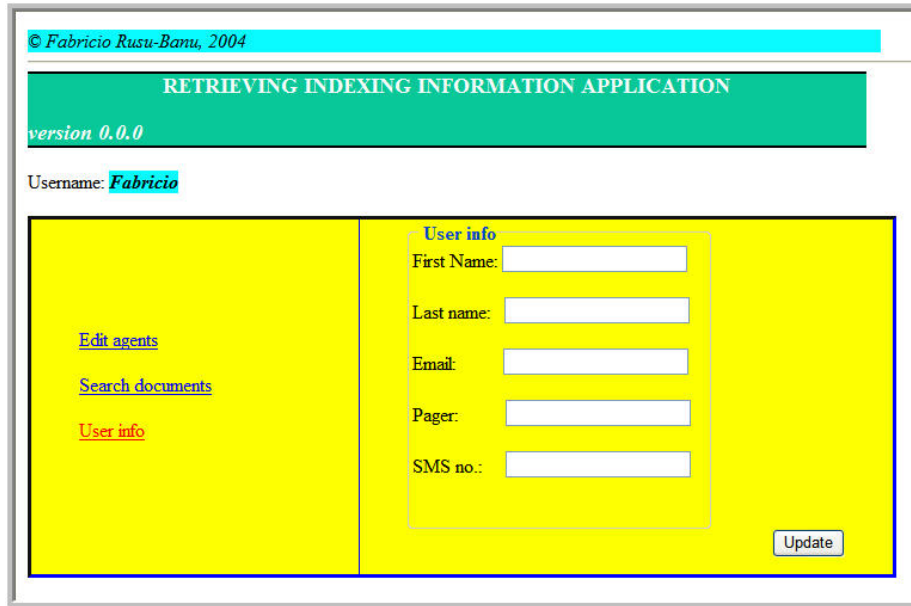


Fig. 6 – Configuring new user interface

5.2. Defining agents

Everybody can agents as many he wishes (or the Administrator allows that). To ad a new agent, click on “Edit Agent” and edit the following page:

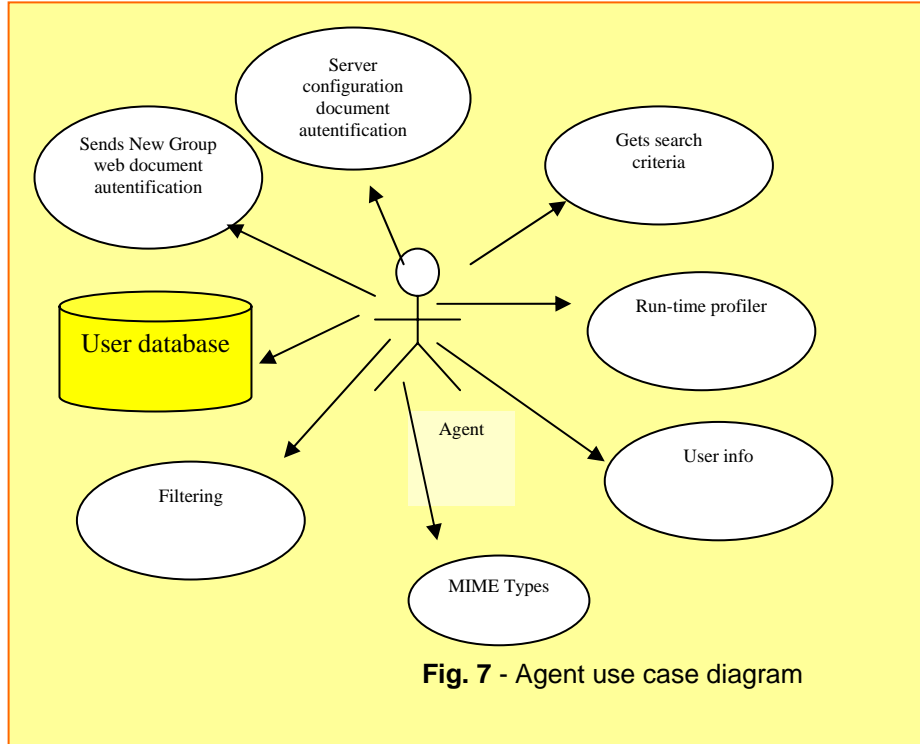


Fig. 7 - Agent use case diagram

Fig. 8 – Creating/deleting agent interface

The user basically has to do not so many things. Firstly he has to enter a name to the new agent. Then he has to design a query, which is a word or a phrase. The threshold is a minimum score for the document returned by the agent. The user can establish the method of delivery as its convenience. The last step is to click “Create Agent” to add its new agent to the agent repository. To modify the respective agent you need to change the agent properties. The list of all agents used by the respective user are found in the Agent Repository where there are stated their properties.

Here are some brief hints about the coding process:

```
class RIIAgentNew {
/* this class defines a new agent */
int RIIARepository (repository); //the address of an agent handle
char RIIAgent* (agent);
char RIINewArgRec*(newAgent); //a pointer to a RIIAgentNewArgRec
//structure }
```

```
class RIIAAgentSetInfo {
/* this class changes information associated with a particular agent */
int RIIAAgent (agent); //agent handle
char RIIAAgentSetArgRec* (setArg);      }

class RIIAAgentGetInfo {
/* it provides information about an agent
int RIIAAgent (agent);
char RIIAAgentGetArgRec (*getArg);
char RIIAAgentGetOut * (getOut); //the address of the getOut handle.}

class RIIAgetInfoFree {
/* it frees an RIIAAgentGetOut handle */
char RIIAAgentGetOut (*getOut)      }

class RIIASearchNew {
/* it searches for agents in an agent repository*/
int RIIARepository (repository);
char RIIAAgentSearch (search);
char RIIASearchNewArgRec* (newSearch); }
```

6. Conclusion

The *Retrieving Indexed Information Application* (RIIA) is an agent software application having its purpose to gather documents from Internet or Intranet automatically. Precisely many agents configured particularly in accordance with specific jobs, indexes, area of interests retrieve information from whatever sending automatically to the user in web page format (News Group Info), email, pager or SMS) the results returned. This is a typical agent software application where the computer does the job alone instead spending the user's time and resources in identifying, analyzing and retrieving information accordingly with the user's field of interest.

BIBLIOGRAPHY

1. **Behrouz H. Far** - SENG 609.22 Agent based software engineering, Course Notes and Distribution Materials, University of Calgary, 2004
2. **[AgentBuilder R.M.]**- An Integrated Toolkit for Constructing Intelligent Software Agents, AgentBuilder, Reference Manual, April 2000
3. **[AgentBuilder U.G.]** - An Integrated Toolkit for Constructing Intelligent Software Agents, AgentBuilder, User's Guide, April 2000
4. **Weiss, G** - "Multiagent Systems", MIT Press, 1999
5. **Etzioni and D.S.Weld** – Intelligent Agents on the Internet: Fact, fiction, and forecast", IEEE 1996
6. http://agents.umbc.edu/Applications_and_Software/Software