

Personal Information Agent System Design

Sergey Smirnov

Hasso-Plattner-Institute for Software Systems Engineering
P.O. Box 90 04 60, 14440 Potsdam, Germany
sergey.smirnov@student.hpi.uni-potsdam.de

ABSTRACT

Today the stream of information is overwhelming. People can not effectively find the information they need manually any more. They need intelligent and personalized tools to filter the needed information – personal information agent. However there are a lot of possibilities of personal information agent design and implementation. The user requirements and the selected document model drive the design and the implementation processes. This paper proposes the system design of the personal information agent, outlining the main system components, their tasks and interaction between them. It also tries to estimate which technologies are applicable for the implementation of the proposed design.

Categories and Subject Descriptors

D.2.11 [Software Engineering]: Software Architectures --- *Domain-specific architectures*; H.3.4 [Information Storage and Retrieval]: Systems and Software --- *Distributed systems*

General Terms

Documentation, Design.

Keywords

Personal Information Agent, System Design, eTVSM.

1. INTRODUCTION

Since the 1980s years the amount of information which is stored in the digital form has been growing all the time. The rapid development of hardware has been providing possibility to store more and more information. The Internet allows to access data which is stored anywhere in the world. It seems that everybody has unlimited access to the information. However, the quantity of information is so huge that a human being is unable to work effectively with it. People need assistance to get the information they need from this ocean of data. There are search engines, which help to get the information the person wants to get. Unfortunately, this solves the problem only partially.

In the recent years there were a lot of attempts to create a personal information agent [3, 4, 5, 8, 9, 16]. First the term personal information agent (PIA) must be defined. Let us use the definition given by [15].

Personal information agent “*is a computational software entity that may access one or multiple, distributed, and heterogeneous information sources available, and pro-actively acquires, mediates, and maintains relevant information on behalf of its user(s) or other agents preferably just-in-time*”. This definition is given by the group which does research in the fields of software agents. Nevertheless, it defines the main features of the personal information agent regardless the technology which is used for the implementation.

This paper proposes the design and some implementation issues of the PIA. As soon as the design process is driven by the requirements they are formulated in Section 2 first.

The PIA is software system that performs the information filtering task – *the selection of documents from a dynamic stream of documents using some kind of static profile* [1]. The concept of the user profile and its system representation is introduced in Section 3. This section continues with the ability of the system to adapt to the changing user interests using the profile.

Every information filtering system is based on some document model. Document model represents natural language documents. There are several document models available. Section 4 starts with the question of document model choice. To make an appropriate choice criteria are selected: implementation possibility, performance, linguistic features support, document similarity representation and simplicity of the model. According to these criteria section 4 proposes enhanced topic vector space model (eTVSM) [1] as the document model in the PIA.

The document lifecycle is described in section 5. It clears what happens to the document within the system and provides additional information for further design process.

In the first design phase the high level architecture of the PIA is developed. The main components and their tasks are defined. The interaction between the PIA components is described. These issues are addressed in section 6.

Finally, implementation issues are discussed in section 7. Usage of the database management system in the implementation is described. The choice of the Java 2 technology as the basis for the implementation is explained.

2. REQUIREMENTS

The design process of applications is driven by the user requirements. That is why it is very important to define the requirements in the very beginning. Summarizing the definition given by [15] and the requirements mentioned in [16] it is possible to formulate the requirements for the PIA. In this section user and additional requirements will be discussed.

2.1 User Requirements

The task of the PIA is to filter information, regardless of its source type. That is why the PIA must be able to work with different types of information sources (e.g. HTML pages, PDF documents, mailboxes, RSS). The search for the new documents must be autonomous, which means that it does not need the human control.

The PIA must provide an easy access to all the documents in the system. This requirement implies that once the document is available in the information filtering system it should be always available to the user. This aspect of the problem is directly connected with the system design of the PIA. Another issue is closely coupled with the user interface design: the user interface must ease the user access to all the documents in the system.

Performing the filtering task the PIA acts on behalf of the user. To do this it needs to know the user interests and according to them provide appropriate documents to the user. User interests are changing. To be helpful for the user the PIA must adapt to the changing interests.

Even if the PIA is able to follow user interests there is no guarantee that it does it perfectly. That is why the PIA should enable a user to express explicitly what he needs. For example, the user should be able to put the document from one document class to another if he finds the classification to be wrong.

A new application installation can become a barrier for potential users of the PIA. To attract more users the PIA must create no barriers. The web interface may help to achieve this goal. Today most of the personal computers have browsers installed. This is enough for using the application with the web interface. That is why the web interface is the most preferable solution in this case.

The last but not least is the PIA performance. Being a rather complex software system, the PIA may perform several tasks at once: looking for and accessing new documents, analyzing incoming documents and providing documents to the users. On one hand these tasks can be computationally intensive and on the other hand must be executed in a certain amount of time. That is why the system performance turns out to be extremely important for the PIA.

2.2 Additional Requirements

The design of the PIA must assume that the development of the product should start from the simple prototype. This prototype must be able to perform all the main tasks of the system. The system architecture should give a possibility to scale the PIA, keeping it simple.

3. USER PROFILE

The main task of the PIA is information filtering. In order to accomplish this task the PIA needs the *user profile*. In this section the user profile concept is discussed: its definition is given, the profile structure is proposed. Finally, the profile is considered as a mechanism that helps the system to adapt to changing user interests.

3.1 User Profile Structure

The PIA is a multi-user system. Each user has his specific interests. The PIA must know about these interests to perform information filtering task. Therefore, the user informs the system about the topics that are of his interest. He can do this explicitly,

by choosing the topics. The user profile reflects user's interest in these topics. The profile is the model of the user interests used by the system. Each user has own profile. The profile consists of *topic notions*: the profile associates a topic notion with each topic.

A topic notion can be initialized by a "query". The query concept is borrowed from information retrieval systems where users write queries to achieve their goals. In fact queries are natural language documents. That is why it is also possible to build document models for them. Thus, topic notion consists of one or more document models.

The important question is what happens when the new user registers within the PIA. Every new user forms the personal profile, filling it with topics. After this the PIA must find documents which are interesting to the user and link them with the profile. During this process the PIA uses topic notions with which the profile was initialized.

Thus, the user profile consists of topic notions. There is one topic notion per each topic declared to be of user interest. Topic notion contains association links to the documents once appeared to be interesting to the user in this topic.

3.2 Profile as the System Adaptation Mechanism

The user may change his interests. The PIA must adapt to this behavior of the user. It means that when the user interests are changed, the PIA must reflect this in the model of the user interests – modify the user profile. However, adapting to the changing user interests is not a trivial task. A certain learning algorithm is needed to modify the user profile improving filtering results.

Since the user has registered in the system, a profile reflecting his interests is created. The PIA uses the profile to provide documents to the user. It is natural for the user that any document provided to him once must be always accessible later on. On the other hand the user profile is changing together with the user interests. This means that the relevancy of the document to the topic notion in the profile is also changing. At some point in time the document can be found irrelevant to the topic notion and become unavailable to the user. To avoid sudden "vanishing" of any document the profile must store the association link between the documents once appeared to be relevant and the topic notion.

4. DOCUMENT MODEL

One of the milestones in the design process of the personal information agent is the document model choice. In this section the criteria for the document model choice are formulated. Afterwards, it is shown that eTVSM corresponds to the mentioned criteria and it is proposed as the document model. The main features of eTVSM are described.

4.1 Document Model Choice

Every information filtering system is based on a document model. The document model has a great influence on the system architecture and system properties. The document model must fit the system architecture. Otherwise it is very difficult to achieve desired properties of the system, e.g. performance or scalability. The wrong choice of the document model can hardly be compensated by other components of the information filtering system.

To make a choice which document model to choose criteria are needed. The presence or the lack of the following features of the document model can become the criteria which help to make the choice.

First of all it must be clear how to implement the document model. It assumes, for example, that the model describes how to obtain information about similarities between terms. The document model used in the system should provide good performance. This means that there must be a possibility for performing computations in the efficient way. The document model must have a broad support for linguistic features. If the document model supports many linguistic features it increases the overall system performance and improves the quality of filtering results. Different tasks performed by the personal information agent are based on obtaining the similarity between two documents. That is why it is very helpful when the model can clearly express the similarity between two documents. The example could be the similarity which ranges between 0 and 1. However, not all the document models give this possibility. The last feature of the document model is its simplicity. If the document model is simple to understand it is easy to work with it, design and implement the system using this model.

4.2 eTVSM

In this paper eTVSM is proposed as the document model. It satisfies most of the criteria mentioned above. eTVSM provides a formal mechanism how to obtain term similarities. There is a clear way how it can be implemented. [1] shows the way how to implement the information filtering system based on eTVSM in order to achieve high performance. eTVSM is able to represent homography, metonymy, word groups and many other linguistic phenomena. The document model integrates stemming and stopwords removal. The presence of these features in the model improves the performance, as soon as no additional preprocessing steps are needed. As well as its predecessors, VSM and TVSM [17], eTVSM provides the document similarity between 0 and 1. eTVSM has many different features, but it is relatively easy to understand its main principles inherited from vector space model (VSM). There is a lot of design and implementation techniques for VSM, which can be useful for eTVSM implementation [2]. According to the eTVSM features listed above it is possible to say that this model can provide the high system performance and dramatically improve the quality of information filtering.

Let us look at the main principles of the eTVSM. The heart of the eTVSM is an ontology. The ontology can be represented by a topic map [1]. Topic map is a set of topics with the given relations between them. Each topic τ_i from the topic map has a corresponding topic vector $\vec{\tau}_i$. All the topic vectors are lying in the space S . The space dimension is equal to the number of topics in the topic map. There is a mechanism for obtaining topic vectors from the topic map [1].

However, a method for representing natural language documents in eTVSM is needed. Each document in the eTVSM is an ordered list of words. To obtain a document model for a document several steps are needed: words are assigned to word-stems, word-stems are assigned to terms and terms are assigned to interpretations. The result is that the document is represented with the list of interpretations. Interpretations represented in the space S with interpretation vectors. The interpretation vector can be calculated

using the topic vectors. Each document has a document vector. The document vector can be calculated using interpretation vectors and the number of their occurrences in the document.

The document similarity in eTVSM has the following properties:

1. $sim(k, l) \leq 1$,
2. $sim(k, l) \geq 0$,
3. $sim(k, l) = 1$ if $k=l$,
4. $sim(k, l) = sim(l, k)$,

where k, l – documents and $sim(k,l)$ is the similarity function.

5. DOCUMENT LIFECYCLE

At this point it is necessary to describe the lifecycle of the document within the system. The document lifecycle is the sequence of events associated with the document in the system and operations which can be done by the system with the document.

The PIA searches for the new documents in the environment. Since there are a lot of heterogeneous document formats, the found document should be transformed into one system unified document format. Afterwards the document model is built. The document (using document model) can be classified into some topic. The next step is to check which users are interested in the document. User profiles help to do this. If some user appears to have interest to the document then an association link between this document and the appropriate topic notion is added to the profile. The existence of the association link shows that the document was or is in the scope of the user interest.

6. PERSONAL INFORMATION AGENT COMPONENTS

This section describes the PIA architecture. The usage of the component model is motivated. The main components are identified: their names and tasks are given. Afterwards, for each component a detailed description is provided.

6.1 Architecture Outline

From the definition given above it is clear that the PIA maintains several tasks. All the tasks have a common goal. The goal determines existing information dependencies in the system. One of the tasks of the paper is to provide a system design that helps to manage the information dependencies. The component model aids to accomplish this task. Each component maintains certain task and encapsulates the corresponding logic. Certain interfaces are defined for the components. These features of component model help to manage information dependencies.

The following components of the PIA can be identified:

- ontology manager,
- document search robots,
- mediators,
- linguistic analyzer,
- data storage,
- document manager,
- user interface,

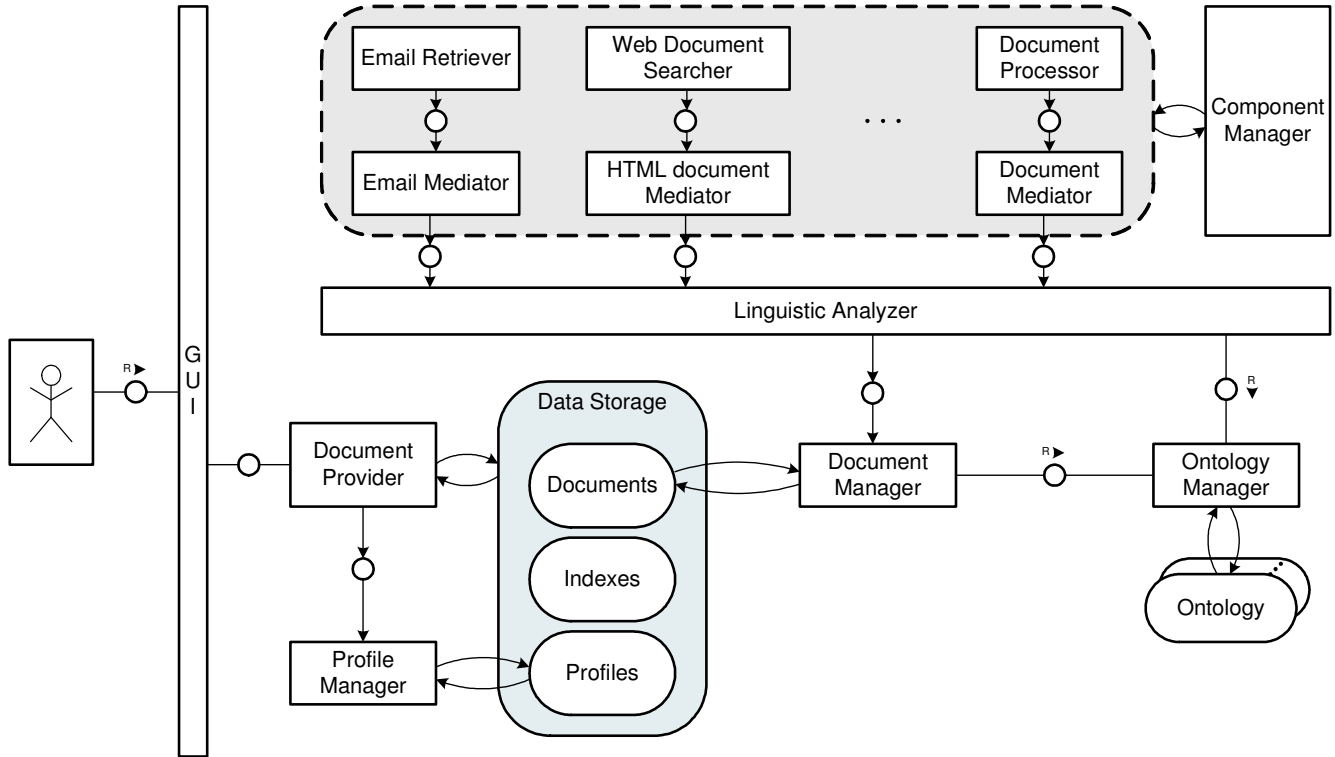


Figure 1. Logical architecture of the personal information agent (FMC notation)

- profile manager,
- document provider.

The components of the PIA are shown in Figure 1.

The document search robots access information sources of certain type and extract documents from those sources. Mediators translate documents in the specific format into the document in the standard system format. The document in the standard system format is the natural language document. The linguistic analyzer creates document models for the natural language documents. To accomplish this task the linguistic analyzer needs an ontology. That is why it interacts with the ontology manager. The ontology manager provides a standard interface to the ontology for the system components. After the document model is available the document manager makes a decision how it should be stored. It interacts with the data storage that keeps the documents in the system. The user interface component provides an interface to the PIA user. The user interface presents the documents to the user. To accomplish this task it communicates with the profile manager and document provider. The former has to provide a standard interface to the user profiles for the system components. The latter extracts documents from the data storage.

In the remaining part of the section the named components will be presented in details.

6.2 Ontology Manager

The intention to use eTVSM as a document model leads to the necessity to work with an ontology. However, the choice which

ontology to use is not obvious. During the first period of the lifecycle it makes sense to use a simple ontology, which can be developed later on. In the initial variant of the ontology all the terms can be orthogonal and during the system evolution term relations can be added. Another variant could be the usage of topic maps. It is a more sophisticated and laborious variant, but it can bring better performance to the system and better fits the idea of eTVSM. Another step is to use an existing ontology like WordNet [18]. This approach is the most promising: WordNet already includes a lot of terms and relations between them. On the other hand it is not clear how exactly WordNet can be used by eTVSM. The relations between terms in the topic maps and WordNet are similar. However, a research in this area has to be done.

The listed alternatives show that the PIA must be able to work with different ontologies: an ontology must be replaced by another one if it is necessary. This replacement must be transparent to all the other components of the PIA to minimize modifications in the system design. Transparency can be achieved if a standard ontology interface is provided to all the system components. Thus a component *Ontology Manager* is introduced. Since then all the components which need to interact with the ontology have to interact with the ontology manager. The ontology manager must be able to understand requests coming from the system components and transform them into actions upon the current ontology. After receiving results from the ontology, the ontology manager must transform them in a format understandable by all the other PIA components.

6.3 Document Search Robots

The system must have an ability to get new documents from the environment. At the present moment there are a lot of types of information sources in the internet: web pages, mailboxes, RSS, different kinds of subscriptions. Each of these sources has specific features: a format of documents provided and an access method. To cope with the environment complexity, different types of robots can be designed. The task of the robot is to be able to perform search through the special kind of information source. For example, one robot can specialize on searching for new HTML documents and the other one – on searching for messages in mail boxes. That is why it must know how to access the information sources of special type and how to pull documents from them. After the document is obtained it must be tagged with its location and provided to the mediators. The information about the location of the document must be enough to access the document when it is needed and stored together with the document in the system.

The lifecycle of the document search robot is presented in Figure 2. At first the robot is initialized. Then it tries to check if there are mediators of the certain type available. If there are no mediators available the robot is destroyed. Otherwise, it stores the “addresses” of available mediators. This means that the robot can perform its task and is active. At some point in time the robot can be stopped - destroyed.

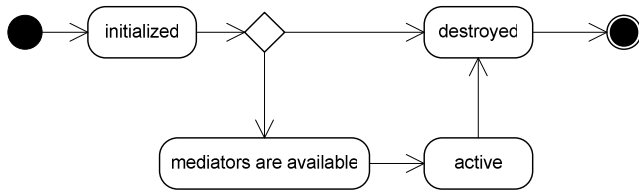


Figure 2. Document search robot lifecycle. (UML Statechart Diagram)

6.4 Mediators

Mediators must be capable to understand the special document format (e.g. HTML documents or PDF documents) and transform the document in this format into the document in the unified system document format.

The robots must know how to communicate with the mediators which can process the format of the founded documents. It means that the robot that perform search through HTML documents can work with several mediators. Anyhow, these mediators must be able to process HTML documents. The other way round, the mediator can receive documents from several robots. Again the document format which they work at must be the same. On the other hand, robots and mediators that work on different types of information sources and document formats are independent. Robots interact actively with the environment and send their results to the mediators. That is why it is possible to locate each robot on a separate machine. This will provide maximum performance and may also address some security issues.

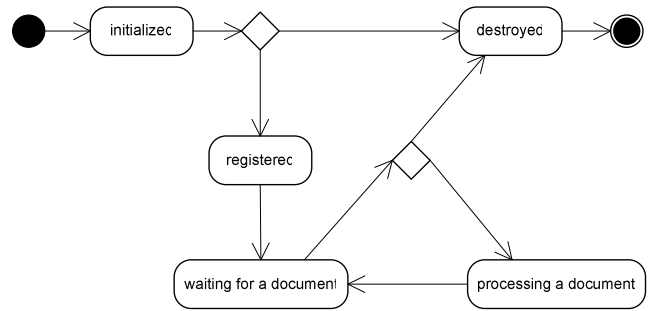


Figure 3. Mediator lifecycle. (UML Statechart Diagram)

The lifecycle of the mediator is presented in Figure 3. After the initialization the mediator registers with the configuration manager. After registration the mediator waits for an incoming document from the robots. Otherwise, it is destroyed. When the mediator is waiting for the document it can be destroyed (by the component manager). However, if it receives the document to be processed, it can not be destroyed until the task is executed.

The configuration of this part of the system is rather laborious: it must be possible to add new components (robots and mediators), remove the old ones, inform robots with which mediators they must communicate and perform other administrative tasks. Component manager can help to solve these tasks, automatize some of them (e.g. mediator discovery by robots).

The output of the mediator is the document tagged with its location and the content of the document in the unified system format.

6.5 Linguistic Analyzer

The task of this component is to build a document model for each incoming natural language document. To achieve this it has to perform linguistic analysis of the incoming document. Mediators provide documents in the unified format (for example, plain text or formatted text). The eTVSM as the document model drives the linguistic analysis process. The main steps are:

- assigning words to documents (considering position of the words in the document),
- mapping words to word-stems,
- mapping word-stems to terms,
- mapping terms to interpretations,
- mapping interpretations to topics.

Each of these operations is a separate task. It can be represented as the pipeline: every element receives the output from the previous transformer as the input (Figure 4).

The independence of each task from the other gives great possibilities for scaling. It is possible to perform transformations of one type in parallel (together with the load balancer to manage them) or even perform each mapping on a separate computer.

Mapping terms to interpretations requires ontology usage in order to find establish relations between interpretations and topics. At this level some linguistic phenomenon can be resolved, e.g. homography and word groups.

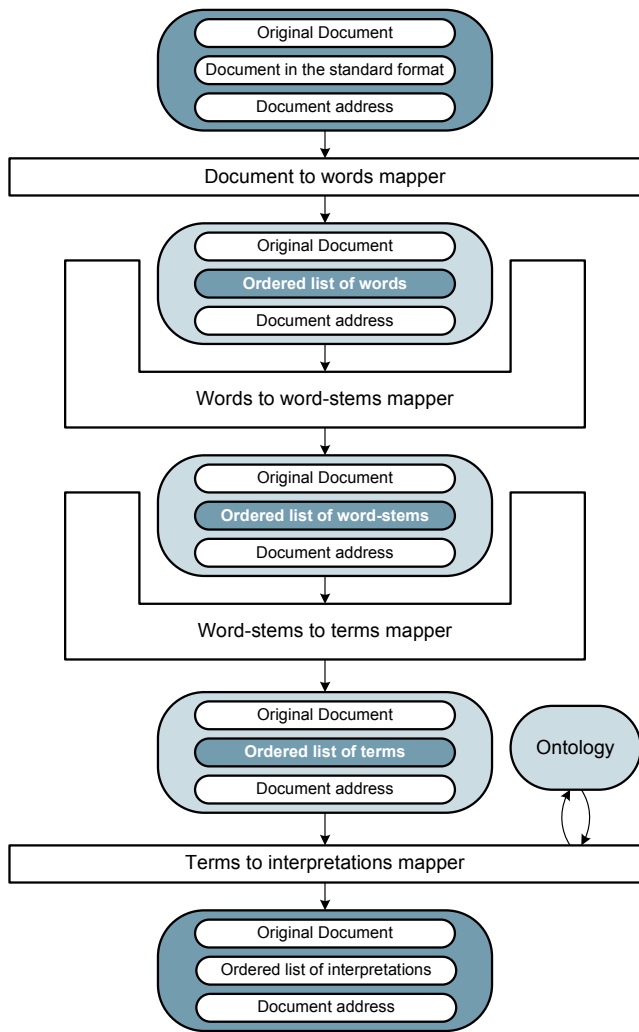


Figure 4. Linguistic analysis steps.

The output of the linguistic analyzer is the document tagged with its location and the document model.

6.6 Data Storage

At this step we have a document model available. To be able to use it later on we have to store it. It is not clear yet how to organize the component, responsible for document storing. The eTVSM together with the document lifecycle drive the way the document model is stored. The general eTVSM ideas as well as the document lifecycle have been already discussed.

The first task to be solved is to classify the document to one or several topics on the basis of its document model. The classification is helpful, since the classification of one document is performed only once. Otherwise search through all the documents is executed many times. When the document classification is available it helps to narrow the area of the search.

To classify the document d a classification algorithm is needed. For example, k-nearest neighbor algorithm [19] can be used. The

following steps have to be done. The document vector \vec{d} has to be obtained for the document d . Vector \vec{d} lies in the space S together with the other document vectors which have already been classified. Applying the k-nearest neighbor algorithm it is possible to put the incoming document into the appropriate topic.

The classification algorithm can also be used to determine if the incoming document is relevant to any topic notion in the user profile. If it occurs that the document is relevant than an association link between the topic notion and the document is created as it has been described above.

The document storage techniques need the document similarity to be available. To obtain the document similarity the interpretation vector similarity and occurrences of interpretations in these documents are needed.

To store the number of interpretation occurrences in the document the technique of inverted indexes can be helpful. Inverted index is widely used in information retrieval systems [2]. In the original approach the inverted index is a pair, consisting of a term and a list of documents associated with it. This pair can be used for increasing the performance of search. In the PIA the idea of inverted index can be used to store pairs of interpretation and list of document models, where this interpretation occurs.

Similarity between a pair of interpretation vectors is the same for all the document vectors in assumption that ontology does not change. The interpretation vectors depend on the ontology.

Every interpretation vector can be defined in the eTVSM with the help of the interpretation weight and the set of topics to which the interpretation is related. The PIA must store this information for every interpretation vector. When the ontology is updated it helps to obtain a new value for the interpretation vector. Changes in the ontology may also modify the set of topics to which interpretation is related.

If the ontology is changed some interpretation vectors should be recalculated. The decision which interpretations have to be changed depends on how the ontology changes. Assume the ontology is the topic map. If a new topic τ_k is orthogonal to all the topics which were in the ontology before, then it has a relatively low influence on them. In this case only the dimension of the space S is increased. If an added topic τ_k is a leaf in the topic map, then it influences all its super topics. This leads to the necessity of recalculating topic vectors for all the topics, which are τ_k super topics. Moreover, interpretation vectors, which depend on the changed topic vectors, are also changing. It shows that every ontology update can be very expensive in the sense of calculations following it.

6.7 Document Manager

After linguistic analyzer has provided the document model it has to be stored in the way described in the previous section. It assumes that there is a component which performs analysis of the document model providing certain items needed for storing the document. The optimization of the document storage methods is based on the content of the document. That is why document manager has to have an access to the ontology. It means that it

communicates with the ontology manager. The results of the document manager work are kept in the data storage.

6.8 User Interface

As it has been already mentioned above the PIA is a multi-user system. It runs on a remote server providing certain facilities to the user. To be able to use these facilities the user must have a client running on his machine. There are different possibilities how to distribute the tasks between the user client and the server. Each of them has its own advantages and disadvantages. If the client is thick then it can provide good performance and more sophisticated facilities. On the other hand, the thick client needs additional installations on the user machine. The thin client is simpler. It can be designed in such a way that users can avoid any additional installations or minimize them. To increase the number of users the PIA must not create additional barriers for the potential users. As soon as many users tend to avoid additional installations the thin client is a preferable choice. The client side of the PIA should contain only user interface to be simple and minimal.

In order to ease the interaction with the system for the user the PIA should provide a graphical user interface. At the present moment web interface can effectively solve the problems discussed. It is powerful enough to create a GUI for the PIA. It needs only a standard browser to be installed on the user machine. It is not very laborious to implement it.

The process of interaction between the user and the PIA involves GUI component. With its help the user obtains documents which are interesting for him. In order to accomplish this task the PIA has to “learn” what is interesting to the user. The web interface component has to collect information about user interests [8, 9]. Thus the GUI main goals are representing documents to the user and collecting information for the user profile. These activities assume that GUI component interacts with the *document provider* and the *profile manager*.

6.9 Profile Manager

The profile manager is the component of the PIA which is responsible for maintaining the user profiles through its lifecycle. This assumes that the profile manager creates the profile of a new user, adds new topics to the profile and deletes topics if it is needed. It also controls the modification of topics: which documents are relevant to the topic and which are not. The profile manager provides a standard interface to all the other components helping to abstract from the implementation details.

6.10 Document Provider

Document provider is the system component which finds appropriate documents in the data storage for the user, basing on the user profile. The way document provider works can be described in the following use case. User signs into the system. He does this with the help of GUI. The GUI has to show him relevant documents now. It requests document provider to prepare a list of documents for this user. Document provider requests the profile manager about the user interests. After receiving the answer it prepares the list of documents for the user and proposes it to the GUI. This interaction is presented in Figure 5.

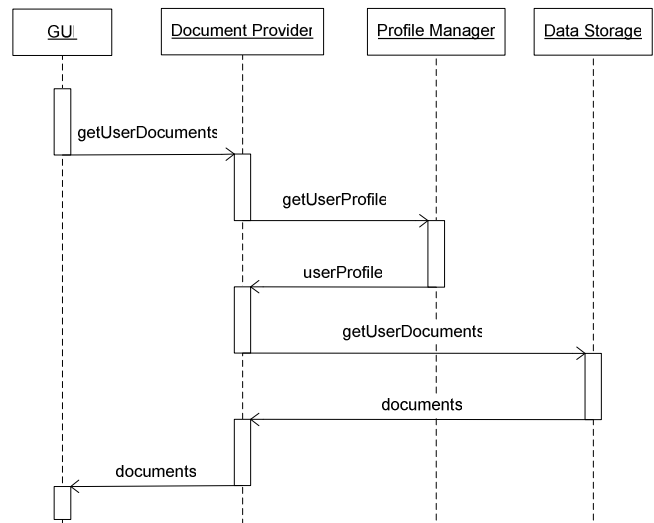


Figure 5. GUI interaction with other components (UML Sequence Diagram)

In this section the main system components and their responsibilities have been described. The communication channels between the components have also been defined.

7. IMPLEMENTATION ISSUES

In the previous section the general architecture of the PIA has been discussed. However, only the outline of the system architecture has been provided. The question which technologies can be useful for the PIA implementation is still open. This section describes how DBMS can be used in the PIA. The advantages and disadvantages of different implementation methods for an ontology storage are argued. Finally, the problem of technology choice is considered.

7.1 Application of DBMS in System Implementation

The PIA has to process large amounts of information. It stores a lot of items such as documents, document models and user profiles. With a certain period of time new documents are coming. They must be processed and stored. Later on these documents should be quickly accessed. In other words the PIA must have an effective information storage. At the present moment database management systems (DBMS) successfully solve this task. There are different types of DBMS: relational, object-oriented or hierarchical. In this paper relational DBMS (RDBMS) are proposed to be used as the tool for storing data. They are well known and simple to use. Although other types of DBMS can provide better performance for specific data types, RDBMS has better performance for different data types in general [6]. In [1] relational DBMS was proposed for storing the data in information filtering and retrieval systems. It has been shown how it is possible to use the relational model for storing documents and their models.

The usage of DBMS gives additional advantages. It can help to implement the PIA. Many of the modern DBMS support caching

and clustering mechanisms. Caching may help to address the performance requirements and clustering – scalability of the PIA.

The DBMS caching mechanisms can be helpful to implement some typical PIA tasks. The tasks deal with the same data, are regularly repeated and are intensive in the sense of calculations. The example is the relevance checking of the new document to the user profiles. Every of these tasks can be implemented manually with the help of a programming language. Another approach is based on delegating the task to the DBMS used in the PIA. In the former case a good algorithm has to be chosen and many performance issues have to be addressed in the algorithm. The advantage is the full control over the algorithm: it is clear what algorithm is used and what happens inside it. In the latter case, DBMS use its internal mechanisms for performing the calculations. The main advantage is the high system performance together with the relative simplicity.

The DBMS allows scaling. Most of the DBMS provide a possibility to create clusters. The scaling on the level of DBMS simplifies the design of the system: the clustering is maintained by the DBMS. The DBMS clustering is transparent for the other PIA components.

The DBMS features, such as caching and scaling, ground the usage of DBMS in the PIA.

7.2 Storing the Ontology in DBMS

In [1] the RDBMS was used for storing the ontology. This approach provides high performance: a lot of operations on the documents and their models can be delegated to the DBMS. However, this paper proposes a component for accessing the ontology – ontology manager. Introduction of the ontology manager may lead to lower performance in comparison to the approach discussed in [1]. On the other hand, it is easier to use different ontology storage methods and replace ontologies.

Anyhow, relational DBMS still can be used for storing the ontology. The main principles, described in [1] can be preserved.

There is another important outcome of the ontology manager introduction. To optimize the system performance the PIA stores additional information. This information can be used for determining if the document is relevant to the user profile or obtaining the similarity between two documents. This additional information is based on the ontology used and depends on it very much. The presence of the ontology manager in the system leads to the necessity of clear separation between this additional information and the ontology itself. Another important aspect is the consistency between the ontology and the additional information. The consistency must be supported by the document manager. The document manager maintains the process of storing documents and their models, using the ontology. If it is needed it creates additional information, which is based on the current ontology.

The producers of some existing ontologies, e.g. WordNet, provide their own tools for accessing the ontology. The usage of these products is possible as soon as the access to the ontology is controlled by ontology manager. Anyhow, to address the performance issues it is better to avoid this approach. If the mapping from the WordNet to the topic map is available it is always possible to put the ontology data into the DBMS.

7.3 Implementation Technology Choice

The other important issue in the implementation of the PIA is the choice of the programming language and technology. Today several solutions from different vendors exist, which can be used for the implementation of the proposed design. To do the choice criteria are needed. There are a lot of factors which drive the choice process. Let us choose the most important. The first criterion is the ability of the technology to support the needed functionality. The second criterion is the final product cost. The last one is the performance of the PIA.

The functionality of the PIA is distributed among the components. The low coupling between components provides more freedom in the implementation of each component. The clear interface definition makes it possible to implement different components with different technologies, but the simplicity requirement makes the implementation with one technology preferable. Thus the technology used for the PIA implementation must provide a possibility to implement all the PIA components. This means that the technology should have appropriate application programming interfaces (APIs). Another important point is the scalability requirement. The application scaling should be transparent.

The other important issue when the technology is chosen is the final product cost. The great part of the PIA cost is the cost of applications, which it uses (e.g. application servers and database servers) and the technology on which these applications are based. To minimize the cost of the whole product the cost of these components has to be reduced.

The PIA performs a lot of computations. It must be able to process user requests, new incoming documents and maintain user profiles. That is why its performance is extremely important.

The first and the last requirements make us to have a look at the enterprise platforms such as Microsoft .NET and Sun Java. These products are rather mature. They allow to create a simple system with appropriate performance. The system can be scaled later on, preserving the performance and the proposed design. However, Microsoft solutions are not free. The licenses for the operating system and application servers have to be bought. Enterprise solution based on the Java technology allows cost reduction. It can be deployed on a free operating system. There are free application servers available [13, 14]. That is why Java technology seems to be the best candidate for the PIA implementation.

Some components of the PIA need specific APIs for accomplishing their tasks. It is important to say what APIs provide Java technology for them.

The PIA provides web interface for the user. Java 2 Enterprise Edition includes Java Servlet and Java Server Pages technologies [10, 11], which are able to support HTTP communication. They allow creating a graphical web user interface.

Several PIA components interact with the DBMS. Java 2 Standard Edition includes JDBC API for working with databases [12].

There are some independent open source projects which can be helpful for the other components implementation. The ontology manager works with topic maps. Topic Maps For Java is an open source project [7], which aim is to provide an application programming interface for working with topic maps in Java. Within this project several tools are also provided. The core tool, TM4J Engine, is able to work with topic maps and store topic map

data in a relational DBMS. The following DBMS are supported: MySQL [20], PostgreSQL [21], MS SQL Server [22] and Oracle [23]. TMNav is a desktop application for navigation through topic maps and editing them.

8. CONCLUSIONS

This work proposes the system design of the personal information agent. Initially the user and additional requirements have been defined. The requirements together with the document model have been driving the design process. The high level architecture outlines the system components. The task of each component is described. Interactions between the components are fixed. On the other hand this architecture can be implemented in different ways. That is why in the following part of the paper possibilities for its implementation are discussed. The applicability of DBMS in the implementation of different components is considered. Different aspects of the technology choice are mentioned. The choice of the Java Technology is grounded.

Providing the general architecture of the PIA this work can be the starting point for further research in the field of PIA system design.

Firstly, the design of each component should be specified. This assumes identification of the main classes and interfaces within the components. After this the first system prototype can be developed. The system prototype will show the system parts that have to be improved.

9. ACKNOWLEDGMENTS

My thanks to Dr. Dominik Kuroпка, Artem Polyvyanyy, Lars Trieloff, Raphael Audet and Remi Liance for discussions, ideas and feedback they gave during this research.

10. REFERENCES

- [1] Kuroпка, D. *Modelle zur Repräsentation natürlichsprachlicher Dokumente - Information-Filtering und -Retrieval mit relationalen Datenbanken*. Logos Verlag, Berlin, 2004.
- [2] Singhal, A. *Modern Information Retrieval: A Brief Overview*. <http://net.pku.edu.cn/~webg/papers/IR%20Overview.pdf>, 2001.
- [3] Pannu, A. S. Sycara, K. *A Learning Personal Agent for Text Filtering and Notification*. In *Proceedings of the International Conference of Knowledge Based Systems*, 1996.
- [4] Albayrak, S., Wollny, S., Varone, N., Lommatzsch, A., Milosevic, D. *Agent technology for personalized information filtering: the PIA-system*. In *Proceedings of the 2005 ACM symposium on Applied computing*, Santa Fe, New Mexico. ACM Press, New York, NY, 2005, 54-69.
- [5] Thint, M., Case, S., Azarmi, N., Azvine, B., Hare, S. *Personal information management assistants — from research to commercialization*. Kluwer Academic Publishers, Hingham, MA, USA, 2003, 39-43.
- [6] Date, C. J. *An Introduction to Database Systems*. 7th Edition. Addison Wesley Longman, 1999.
- [7] Topic Maps For Java Project. <http://www.tm4j.org/>
- [8] Joachims T., Mitchell, T., Freitag D., Armstrong, R. *WebWatcher: Machine Learning and Hypertext*. http://www.cs.cornell.edu/People/tj/publications/joachims_et_al_95a.pdf
- [9] Armstrong, R., Freitag D., Joachims T., Mitchell, T. *WebWatcher: A Learning Apprentice for the World Wide Web*. www.cs.cornell.edu/People/tj/publications/armstrong_et_al_98a.ps.gz
- [10] *Java Servlet Technology*. <http://java.sun.com/products/servlet/index.jsp>
- [11] *JavaServer Pages Technology*. <http://java.sun.com/products/jsp/index.jsp>
- [12] *JDBC Technology*. <http://java.sun.com/products/jdbc/>
- [13] Apache Tomcat. <http://tomcat.apache.org/>
- [14] *JBOSS*. <http://www.jboss.com/>
- [15] Special Interest Group on Intelligent Information Agents. <http://www.dbgroup.unimo.it/IIA/index.html>
- [16] Kuroпка D., Serries, T. *Personal Information Agent*. *Informatik 2001 Proceedings*, books@ocg.at, 2001, 940 – 946.
- [17] Vector Space Model. Wikipedia article. http://en.wikipedia.org/wiki/Vector_space_model
- [18] WordNet – Princeton University Cognitive Science Laboratory. <http://wordnet.princeton.edu/>
- [19] k-nearest neighbor algorithm. Wikipedia article. http://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm
- [20] MySQL. <http://www.mysql.com/>
- [21] PostgreSQL. <http://www.postgresql.org/>
- [22] Microsoft SQL Server. <http://www.microsoft.com/sql/default.msp>
- [23] Oracle Database. <http://www.oracle.com/database/index.html>