# A Multi-Agent System for Personalized and Private Service in PDR

Zhou Xingshe, Yu Zhiwen
*Dept. of Computer Science & Engineering*
*Northwestern Polytechnical University*
*Xi'an, China 710072*
*zhouxs@nwpu.edu.cn, yuzhiwen77@sina.com*

Gu Jianhua, Wu Xiaojun, Zhang Yong
*Dept. of Computer Science & Engineering*
*Northwestern Polytechnical University*
*Xi'an, China 710072*
*gujh@nwpu.edu.cn, depender@yahoo.com*

## Abstract

*The overabundance of DTV (Digital Television) programs precipitates a need for new tools to help people find programs and personalize TV viewing experience. In this paper*[*]*, we present a multi-agent system that provides personalized and private service in PDRs (Personal Digital Recorders). This paper details the architecture and components of the system. The multi-agent system observes the users' viewing behaviors in the background, updates the users' profiles continuously, and provides different programs for different users according to their respective profile information. It can protect the users' privacy by means of utilizing XML encryption of the reported information and verifying the user's name and password before reporting his/her profile to service provider. This paper also describes the communication mechanism in the multi-agent system.*

## 1. Introduction

With the rapid growth of communication technologies, there is an overabundance of DTV (Digital Television) programs available for consumers to choose. This precipitates a need for new tools to help people find programs and personalize TV viewing experience.

The TV-Anytime Forum [1] has been developing specifications to exploit mass-market high volume digital storage in consumer platforms. In the context of TV-Anytime, the central element is a new generation of equipment called Personal Digital Recorder (PDR). PDR is a kind of personal digital media storage device that is widely expected to become an extremely popular consumer electronic device for DTV in the near future. Simply put, PDR can store the broadcast TV programs for the user and then recommend them to the user according to the user preference knowledge learned from the user's viewing history.

Given the exponential increase on digital programs available from service providers (broadcaster, Internet, etc), intelligent agent has been given a lot of attention lately. Agents have the distinguishing ability to automate repetitive and time consuming tasks, including learning the user's preference, filtering and recommending programs in the PDR. Besides other features, an intelligent agent has three major features, which are personalized, proactive, and autonomous.

System with only one agent has an overly centralized functionality mode, which may lead to data centralization, and thus hinder the parallel execution of multiple tasks. Multi-agent system (MAS) is the emerging subfield of AI that aims to provide both principles for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors.

There are several advanages to use MAS [2]. First, having multiple agents can speed up a system's operation by providing a method for parallel computation. For instance, a domain that is easily broken into components--several independent tasks that can be handled by separate agents--could benefit from MAS. Second, multiagent systems have the advantage of robustness, which allows the system to have redundant agents. If control and responsibilities are sufficiently shared among different agents, the system can tolerate failures by one or more of the agents. Finally, scalability is another benefit of multiagent systems. Since multi-agent systems are inherently modular, it is easier to add new agents to a multi-agent system than it is to add new capabilities to a monolithic system. Systems whose capabilities and parameters are likely to change over time or exchange among different agents can also benefit from this advantage of MAS.

This paper presents the architecture of a multi-agent system that provides personalized and private service in PDR. It also describes the communication mechanism in the multi-agent system.

[*] Supported by the National Natural Science Foundation of China (60073054).

## 2. Background

### 2.1. Metadata

Generally, metadata is data about data, such as the title, genre, and language of a television program. In the context of TV-Anytime, metadata also includes consumer profile and history data.

For the purpose of interoperability, the TV-Anytime Forum has adopted XML (eXtensible Markup Language) [3] as the representation model of metadata. Recently, XML, which is developed by W3C, has emerged as a standard information exchange mechanism on the Internet. XML allows the encoding of structural information within documents. This information can be exploited to create more focused and accurate profiles of user interests. Furthermore, XML offers many advantages: it allows for extensibility, supports the separation of data from the application, and is widely used.

In our system, the program description and user profile are both represented in XML.

### 2.2. Vector Space Model

VSM (Vector Space Model) [4] is a good method to represent the features of user profile and programs. In VSM, weights are assigned to keywords as importance indications.

In the user profile, there may be a lot of terms, which indicate the user's interests. The terms have weights and orders respectively, in other words, each term is defined as a 3-tuple (term, weight, order). So the user profile can be represented as a vector of these 3-tuples, if there are $m$ distinct terms in the profile, then it will be represented as a vector:

$$P=((t_1,w_1,1),(t_2,w_2,2),\ldots,(t_m,w_m,m)) \qquad (2\text{-}1)$$

$$w_i \geq w_{i+1} \qquad (1 \leq i \leq m)$$

where $t_i$ is a term, $w_i$ is the weight of term $t_i$, $i$ is the order of $t_i$ in the profile. The weights and orders describe the relative importance of the terms in the profile.

For computational reasons, we can take the top $n$ highest weighted terms to represent the user's preference. So the user's profile can be conceptually represented as the following vector:

$$P=(w_1,\ldots,w_n) \qquad (2\text{-}2)$$

where $w_i$ is the weight of term $t_i$ in the profile.

Similarly, a program can be also represented as a vector with $n$ items, which are the same as those in the profile vector, that is term $t_i$ in the profile vector and content vector is the same:

$$C=(u_1,\ldots,u_n) \qquad (2\text{-}3)$$

where $u_i$ is the weight assigned to term $t_i$. The weight $u_i$ is assigned through the following rule: if term $t_i$ is included in the Actors, Genre or Title field of the program's metadata, then $u_i$=2, if $t_i$ is included in the keywords field, then $u_i$=1, otherwise $u_i$=0.

In the classical vector space representation, the similarity between the two vectors of program and profile indicates the degree of relevance between the program and the profile. A commonly used similarity metric is the cosine of the angle between the two vectors. Given a program $C=(u_1,\ldots,u_n)$ and a profile $P=(w_1,\ldots,w_n)$, the cosine similarity can be calculated as follows:

$$sim(C,P)=\frac{C \times P}{\|C\| \times \|P\|}=\frac{\sum_{i=1}^{n} u_i w_i}{\sqrt{\sum_{i-1}^{n} u_i^2 \sum_{i=1}^{n} w_i^2}} \qquad (2\text{-}4)$$

### 2.3. Relevance Feedback

Relevance feedback is an effective and efficient information retrieval technique that can be used to form query vectors based on documents contents [5]. The main idea is to use documents that have already been evaluated by the user, emphasizing the terms that occur in relevant ones and decreasing the terms that occur in non-relevant ones in future formulations of the same query. More formally, when a user has spent a certain amount of time reading a document, the user profile is updated with the following equation:

$$Q_{i+1}=\alpha Q_i+\beta \sum_{d \in R} d-\gamma \sum_{d \in NR} d$$

where $Q_i$ is the initial profile vector, $Q_{i+1}$ is the modified profile vector, $v_d$ is a vector representation of document $d$, $\alpha$, $\beta$ and $\gamma$ are the feedback parameters to be set, and $R$ and $NR$ represent the sets of relevant and non-relevant documents respectively.

### 2.4. XML Encryption

Since XML is the format of exchanging data in TV-Anytime, proper encryption is crucial for the XML data security, particularly user's private data that pass across unreliable networks. XML encryption [6] is being developed by W3C to secure XML data. It enables to encrypt only specified parts of a document, leaving the rest of the document in its original readable form. All recipients can read the unencrypted parts of a document without doing anything, but only authorized recipients can decrypt and read the confidential parts of the document.

There are several benefits to use XML encryption. Firstly, XML encryption brings the advantages of XML to digital encryption, such as human readability and platform independence. Secondly, the ability to encrypt partial documents can save time as well as system resources.

The core element in the XML encryption is the EncryptedData, which is used to transport secret information from the originator to a known recipient. Data to be encrypted can be arbitrary data, an XML document, an XML element, XML element content, or existing EncryptedData or EncryptedKey elements. The EncryptedData element may become the root of a new XML document or become a child element. When an entire XML document is encrypted, then the EncryptedData element may become the root of a new document. When an element or element content is encrypted, the EncryptedData element replaces the element or content in the XML document.

## 3. System Architecture

The system contains five agents; namely, filtering agent, recommendation agent, profiling agent, report agent, and privacy agent. The architecture of the system is shown in Figure 1.

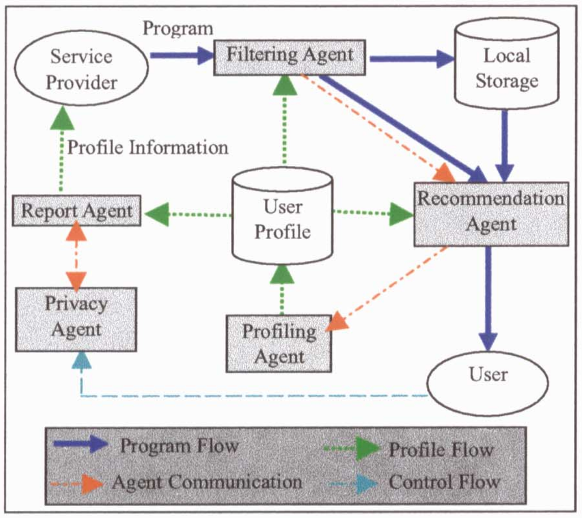

**Figure 1. System architecture**

The filtering agent filters the incoming live program broadcast to PDR, only recording the program that the agent thinks the user would like. The recommendation agent is used to recommend the stored local program to the user according to the learned user preference knowledge. The profiling agent is responsible to update the user's profile according to his/her viewing history. The report agent will report user's preference information to the service provider, who will know what the consumer will be interested in and what the market tendency is through the preference information. The privacy agent will protect the consumer's privacy: doing XML encryption of user's reported preference information, and only when the user's permission is given can it allow the report agent to send the user's preference information to the service provider.

The following subsections describe the filtering, recommendation, profiling, report, and privacy agents in greater detail.

### 3.1. Filtering Agent

When a program arrives, the similarity between the program and the user profile will be calculated. If the similarity is higher than all similarity values of the local programs, the filtering agent will ask the recommendation agent to display the live broadcasting program to the user. Else if the calculated similarity is above the preset threshold $\theta$, we consider that the program is relevant to the user's profile, and then the filtering agent will record the program to the local storage for the user.

### 3.2. Recommendation Agent

The recommendation agent can evaluate the available programs in the local storage using the equation (2-4), and then suggest the $l$ highest similarity programs to the user.

### 3.3. Profiling Agent

The profiling agent will utilize relevance feedback [5] to learn user's preferences with feedbacks from the recommendation agent, and then revise the user profile accordingly.

In our system, when a program is recommended to the user according to his/her profile, and the user has watched it for a period of time, then the user's profile will be refined and revised based on the feedback received. This is done through the modification of the preference terms and their weights respectively. The algorithm is depicted as follows:

For those terms already present in the profile, the term-weights are modified in proportion to the feedback.

$$w_i^{'} = w_i + \alpha \times \beta \times f(i) \qquad (3\text{-}1)$$

$$\beta = \frac{T_r}{T_t} \in [0,1] \qquad (3\text{-}2)$$

$$f(i) = \frac{1}{\sqrt{i}} \qquad (3\text{-}3)$$

where $w_i'$ is the weight of term $t_i$ after the feedback, as well as $w_i$ is the weight of term $t_i$ before the feedback. $\alpha$ is the learning rate which indicates the sensitivity of the profile to user feedback. If the user thinks the feedback is very important and wants the profile learning to be fast, then $\alpha$ can be set larger, other wise $\alpha$ can be set smaller. $\beta$ is the ratio of user's real watching time ($T_r$) to the content's total duration time ($T_t$). $\beta$ can be considered as the user's evaluation to the content which he/she has viewed. In the expression of $f(i)$, $i$ is the order of term $t_i$ in the user's profile, for instance if $t_i$ is the 4-th in the profile, then $f(i) = f(4) = \frac{1}{\sqrt{4}}$ =0.5. $f(i)$ reflects the user's viewing history.

Terms not existing in the profile are handled as follows:

Firstly, calculate the term's weight

$$w_i = \alpha \times \beta \times f(i) \qquad (3\text{-}4)$$

here $w_i$ is the weight of the new term $t_i$, $\alpha$ and $\beta$ have the same meaning as above. Since term $t_i$ is not in the profile before, so $f(i)$ can't be calculated as above, we define it as a default value $\varepsilon$.

Secondly, if the calculated $w_i$ is higher than a preset threshold $\lambda$, we will add it to the user's profile, otherwise discard it, because it is too trivial. $\lambda$ can also be used to control the size of profile and the complexity of computation.

### 3.4. Report Agent

The report agent sends the user's profile to the service provider. Before sending, the report agent will send a message to the privacy agent, asking the user's permission for reporting the profile. If the privacy agent returns "ok", which means permission is granted, the report agent will do its work; otherwise, it will not send the user's profile.

### 3.5. Privacy Agent

The privacy agent will protect the consumer's privacy: using XML encryption mechanism [6] to protect sensitive profile information, and only when the consumer's permission is given can it allow the report agent to send the consumer's preference information to the service provider.

When the privacy agent receives a message from the report agent, it will inform the consumer that the PDR is going to submit his preference information to the service provider, and ask the consumer whether he/she allows this to be done. If the consumer chooses to allow his/her profile to be sent to the service provider, he/she must input his/her password. Then the privacy agent will verify the password, if the password is valid, the encryption will be made for the information that is ready to be reported. The XML encryption implementation or library we used is XMLSec Library [7], which is a C library based on LibXML2 and OpenSSL. The reason we choose XMLSec Library is that it supports major XML security standards proposed by W3C, it is open source, and the manuals to this library are abundant.

Generally, the procedure of our encryption consists of the following steps:

(1) Select which part of the user profile to be encrypted, this depends on the user's tendency.

(2) Select the algorithm (and parameters) to be used in encrypting and select encryption key transport mechanism.

(3) Load the public key of the service provider whom the information will be sent to.

(4) Encrypt the partial data with the public key.

(5) Build the EncryptedType (EncryptedData or EncryptedKey) structure, that is representing the type of the encrypted data, encryption algorithm, parameters, key, etc.

(6) Process EncryptedData. Replace the unencrypted 'element' or 'content' with the EncryptedData element, or use the EncryptedData as the top-level element in a new XML document, or insert it into another XML document for another encoding.

(7) Destroy the temporary objects, such as encryption context, the doc and KeysManager; shutdown XML Security Library, libxml and OpenSSL.

After the encryption has been made, the "ok" message will be returned to the report agent. If the user chooses not to allow his/her profile to be sent to the service provider, or the input password is wrong, the "false" information will be returned to the report agent.

## 4. Multi-Agent Communication

There are five agents in our system. In a multi-agent system (MAS), one key issue is the agent communication. The agents in a MAS need to communicate with each other to achieve something which they can not achieve individually.

An agent communication language should allow agents to interact with each other while hiding the details of their internal workings [8]. In general, agent communication in MAS is based on Knowledge Query and Manipulation Language (KQML). KQML is a language that is designed to support interactions among intelligent software agents. KQML is both a message format and a message-handling protocol to support run-time knowledge sharing among agents [9]. The set of performatives forms the core of the

KQML language. It determines the kinds of interactions one can have with a KQML-speaking agent. Though there is a predefined set of reserved performatives, it is neither a minimal required set nor a closed one. A KQML agent may choose to handle only a few (perhaps one or two) performatives. The set is extensible; a community of agents may choose to use additional performatives if they agree on their interpretation and the protocol associated with each.

In our system, several inter-agent communication pairs exist. For example, the filtering agent need to communicate with the recommendation agent, asking it to display the live broadcasting program when the live broadcasting program is more interesting to the user than all the programs in the local storage; the profiling agent need to communicate with the recommendation agent to get the user feedbacks of the displaying content; the report agent need to communicate with the privacy agent in the authentication process.

After research, we have designed the KQML-based communication performatives for the inter-agent communication in our system. The typical KQML message form in our system is as follows:

```
(performative
   : sender sender-agent-id
    : receiver receiver-agent-id
    : messageid message-id
    : content    ( The Content )
)
```

KQML is a standard language and protocol that intelligent agents can use to communicate among themselves. What is more, the communication and content languages are independent in KQML, this independence affords flexibility. Current KQML implementations have used standard communication and messaging protocols as a transport layer, including TCP/IP, email, Linda, HTTP, and CORBA. But in our system, the agents are programs run on the same host. We can implement them as processes or threads. Since thread has many advantages over process for its sharing memory data, which leads to switching among threads happens much more frequently and efficiently. After research, we decide to implement the agents as threads, and choose UNIX System V message queue as the implementation mechanism for the KQML messages in our system [10].

## 5. Conclusion

In this paper, we propose a multi-agent system for personalized and private service in the TV-Anytime environment. The system is designed to assist users by adapting to their personal preferences, and protecting their privacy. The system has been implemented in the prototype running on PC/Linux. The prototype has demonstrated that our system can really provide personalized programs for users and protect their privacy. With this tool, when a user sits down to watch program there are always some interesting programs to watch in the PDR. Users are no longer slaves to the broadcaster's schedule.

Some improvements of the current prototype are under way. First, we will implement adaptive size of user profile. This can be done by adjusting the value of threshold $\lambda$ (see the details in Section 3.3) while the system is running. It is in fact a problem of tradeoff between storage space and accuracy. Second, we will study and decide the proper value of $\theta$ (see the details in Section 3.1) through experiment. Another improvement is to update the learning algorithm of the profiling agent. We believe these challenges will require more substantial thinking in our system for further research.

## References

[1] TV-Anytime Environment Requirements Document, TV035r6, TV-Anytime Forum, Aug. 2000.

[2] Peter Stone and Manuela Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective", *http://julita.usask.ca/classes/898-02/stone-veloso.pdf*, Jul. 2000.

[3] T. Bray, J. Paoli, and C. M. Sperberg-McQueen, "Extensible Markup Language (XML) 1.0", *http://www.w3.org/TR/REC-xml*, Oct. 2000.

[4] U. Cetintemel, M. Franklin, and C. L. Giles, "Self-Adaptive User Profiles for Large Scale Data Delivery", *Proc. 16th ICDE*, San Diego, Feb. 2000.

[5] P. W. Foltz and S. T. Dumais, "Personalized information delivery: An analysis of information filtering methods", *Communications of the ACM*, 1992

[6] T. Imamura, B. Dillaway, and E. Simon, "XML Encryption Syntax and Processing", *http://www.w3.org/TR/xmlenc-core/*, Oct. 2002.

[7] Aleksey Sanin, "XML Security Library", *http://www.aleksey.com/xmlsec/*, Oct. 2002.

[8] Yannis Labrou, Tim Finin, and Yun Peng, "Agent Communication Languages: The Current Landscape", *http://www.cs.umbc.edu/~finin/papers/ ieee99.pdf*, Mar. 1999.

[9] Finin T, Labrou Y, and Mayfield J, "KQML as an agent communication language", In Jeff Bradshaw (Ed.), Software Agents, Cambridge: MIT Press, 1997.

[10] Uresh Vahalia, *UNIX Internals-The New Frontiers*, Prentice-Hall, 1996.