FAIN: FILTERING AGENT FOR INTERNET

I. A. Badr, M.Sc.
Institute for Automatics and
Software Techniques
Stuttgart University, Germany
email: iman.badr@ias.uni-stuttgart.de

M. M. El-Khouly, Ph.D.
Culture Attache'
Egyptian Embassy, UK.
Corresponding Author: melkhouly@yahoo.com

Abstract- This paper illustrates the design and implementation of FAIN (Filtering Agent for Internet) based on the Gaia methodology and the AUML notation. The proposed system generates and maintains a user profile by learning from examples that are judged and classified by the user during the training stage. A learning algorithm which is based on the Baldwin effect that combines learning with evolution is applied to judge an online stream of documents and filter it according to the learned profile. The illustration of the design and implementation of FAIN has shown the applicability of Gaia complemented with the AUML notation to the modeling of agent-based system.

Keywords: Multi-Agent Systems (MAS), Information Filtering, Internet Applications, Web Programming, Gaia Methodology, AUML.

INTRODUCTION

The old phrase of "needle in the haystack" no longer even begins to suggest the problem of finding pertinent bits of information in the multibillion-page environment we already see on the Web [8]. Traditional search engines have no longer been effective in tackling this problem. According to one of the surveys, no search engine is covering more than 16% of the Web [6]. It follows that, tools automating the search process in a way that relieves the user from trying multiple search engines and formulating his needs have become a must.

Information filtering as Belkin and Croft define [2] is an information access activity similar to information retrieval but differs in that it is concerned with the selection or elimination of text from a dynamic data stream. Information filters are more likely to be personalized to serve the same user's needs over a relatively long period of time. In filtering, removal of data from an incoming stream is based on comparing it to a user profile, which models the preferences of the user. Learning and adaptation are therefore issues of prime importance to filtering systems in order to build and maintain such profiles.

Web agents have recently emerged as a candidate solution to the information filtering problem on the Web. It was predicted that Web agents can really be "the 21st century magic solution to information retrieval problem" [9]. An intelligent agent may be defined as "a system situated within and a part of an environment that senses that environment and acts on it over time, in pursuit of its own agenda and so as to affect what it senses in the future" [5]. According to that definition, an agent application should accomplish the following characteristics [9]:

- Takes *proactive* rather than reactive action to achieve a goal
- Uses *Autonomous*, flexible, goal-defined behavior based on a designed action and rule set.
- Is *social*, in that it can interact and communicate with humans, and perhaps with other agents.

In this paper, a multi-agent filtering application for the Web is proposed and illustrated. The next section overviews the system's functionalities. Then, user interface is illustrated. Following that the general architecture of the system and its agent-oriented design are demonstrated and finally, summary and concluding remarks are presented in the last section.

OVERVIEW OF FAIN

FAIN is an agent-based system, intended as a complementary tool to the traditional search engines. It is intended for users with relatively long-term goals who interact with the search engine in a repeated fashion for each topic in mind. Current search tools don't satisfy the needs and expectations of these users because they depend only on the keywords supplied as input and don't take into account the user feedback for the supplied documents.

FAIN stands as an intermediate layer between the user from one side and search engines and the Web from the other side. It works as a personal assistant that searches on behalf of the user for a certain topic given a set of keywords. The actual role of FAIN starts at the point when the search engine supplies its results. Typically, the search results are in terms of hundreds of thousands which means that it is impractical as well as illogical to attempt to navigate through this huge set of documents.

The search engine results are typically sorted in a descending order starting with these documents having the highest rank of relevance. Practically, users tend to navigate through only those pages appearing in the first few pages or to put it more realistically they only navigate through those pages in the first two or three pages. Documents appearing in the first pages of the search engine are presented to the user for classification in the training stage. The number of documents to be presented to the user

in the training stage is determined by the system but can however be set by the user. In this stage, the system performs no filtering of the search documents and merely accepts user classification in order to accumulate knowledge of the user needs, preferences and interests.

FAIN generates for each topic a stream of documents by crawling search results. It doesn't navigate through just the positively classified documents because of the assumption that interesting documents can be pointed to by irrelevant ones. In this stage the system attempts to emulate the user behavior when skimming a Web page and selectively clicking on only those links whose text seem to be interesting. Selective crawling is achieved by taking into account the user classification along with the link text as displayed in the search results or in the pointing page.

As FAIN automates the navigation behavior of the user, it scans each document it visits and decides based on knowledge accumulated in the training stage that which documents are relevant and worth displaying to the user and which documents should be filtered out. In the preliminary stages of the system use, incoming documents are just classified and displayed to the user for feedback. Later when an acceptable level of trust between the user and his electronic assistant is achieved, documents being negatively classified are automatically filtered out to minimize the level of noise in the output documents. Learning is performed using Naïve Bayesian classifier that has proved its effectiveness in information filtering applications [3,4,10]. Learning, however, doesn't stop at a certain stage but rather starts with learning from examples and continues through learning from feedback.

USER INTERFACE

User interface of FAIN reflects the personal assistant metaphor by displaying an animated character that communicates with the user and tries to relieve him/her from deciding on the sequence of steps for performing a certain task. Microsoft Agent control character has been employed in our system for that purpose. For every topic of interest, the user is prompted to define an initial set of keywords, and possibly a list of starting URLs of favorite documents and sites. Based on these data, a topic agent is created and launched as daemon running in the background to fetch the documents that might be of interest to the user in that topic.

Figure 1 depicts the main screen of FAIN which represents a browser-like interface that is very similar to that of a typical Internet browser like Microsoft IE. Two extra panes on the left are used for displaying the current topics of interest as well as a list of documents suggested for the currently selected topic.

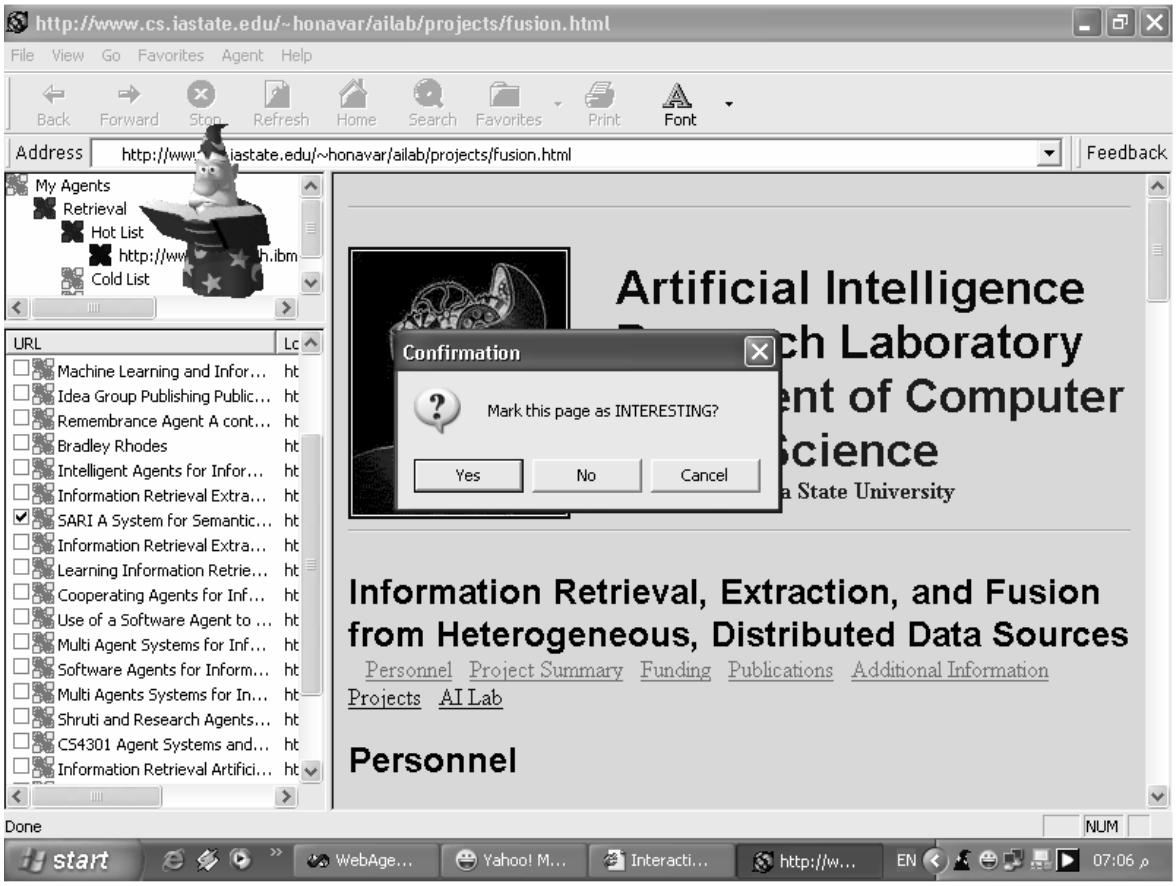

Figure 1: Main screen for the proposed system.

GENERAL ARCHITECTURE

Figure 2 depicts the basic agents of FAIN. As demonstrated in the figure, an interface agent is employed to handle the communication between the user and the system. It is responsible for launching a topic agent upon the user request. The topic agent represents an information agent with the goal of finding relevant documents in a certain topic. It works by creating and managing three agents: the *crawler*, the *DSReader* and the *profile* agent. Each of these agents is delegated a specific task leading to accomplishing the goal of the corresponding topic agent.

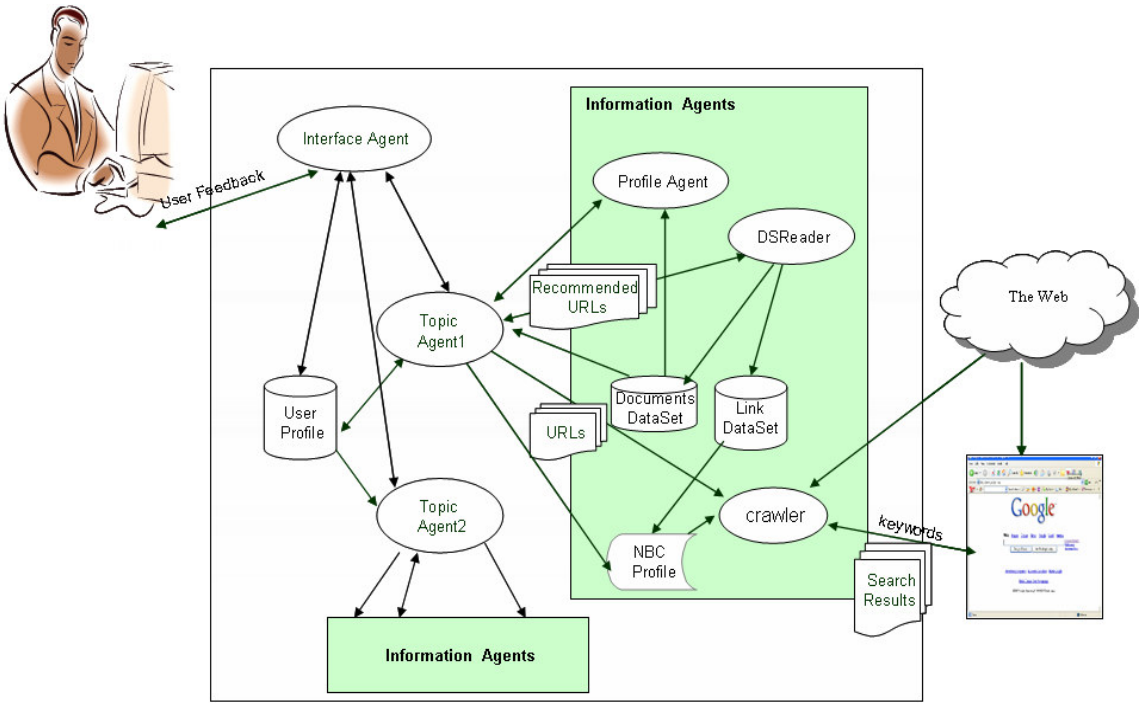

Figure 2: General architecture of FAIN.

The *crawler* is assigned the role of generating a continuous stream of URLs for documents collected from crawling the results attained from the search engine. These documents are believed to contain a set of documents that are relevant to the topic and interesting to the user.

The *DSReader* is responsible for scanning the documents and constructing the datasets required for learning. It scans both documents that were classified by the user during the training stage and raw documents that are directly received from the crawler during the classification stage. Only those documents that were classified by the user are used to construct the required datasets.

The *Profile* Agent is the one responsible for learning and adaptation. It uses the constructed datasets to learn a topic profile using a hybrid algorithm combining Genetic algorithms with the Naïve Bayesian classifier. It is assigned the task of classifying incoming documents and using a user feedback adapts the learned profile accordingly.

SYSTEM DESIGN

In this section, a background on the applied methodology and modeling notation is given followed by an illustration of the design of FAIN.

- BACKGROUND

The design of FAIN has been guided by the Gaia methodology [14]. Gaia was first proposed by Wooldridge et al. [11] and was tuned to support open systems and Internet

applications [12] and was later extended and enhanced by Zambonelli et al. [13]. Gaia represents a role-oriented methodology that regards an agent based system as an organization that is comprised of a set of roles. It proposes but doesn't stick to specific notions and techniques for modeling because according to its developers, it is premature to commit to certain notations at this stage [13].

- ILLUSTRATION

-- ORGANIZATIONAL VIEW

Gaia views an agent-based model as a human organization consisting of a set of roles played by agents. It is a logical view because it is natural to think of agents as human-like entities and to think of an agent-based system as an organization of roles played by these entities similar to a human organization. The first stage in the Gaia analysis tackles the problem of identifying roles of the system, the structure of the organization and the way agents interact with their environment.

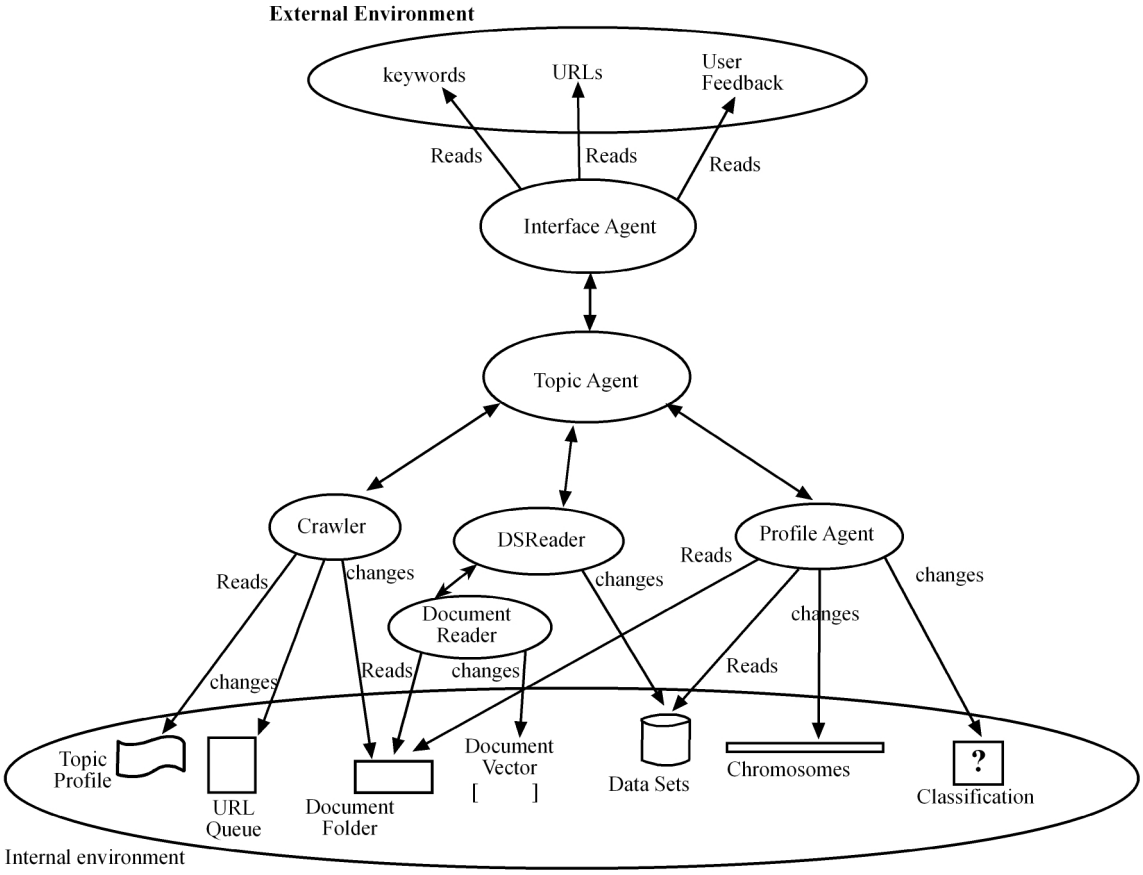

Figure 3: The organizational structure and the environmental model of FAIN agents.

Figure 3 depicts an environmental model of FAIN as recommended by Gaia. It captures the roles to be played by the system agents as well as how these agents interact with their environment to fulfill their expected roles. Therefore, an environmental

model gives an insight about how an agent perceives the environment which is an important aspect of agent development. Furthermore, an environmental model helps in the preliminary determination of inter-agent interactions. As illustrated in Figure 3, we differentiate between an internal and external environment with which FAIN interacts. We consider the external environment to be the resources provided by the user that can be accessed by FAIN. On the other hand, the internal environment represents the Web resources that FAIN exploits.

-- AGENT ROLES

The roles identified in the first stage of the modeling process as depicted in Figure 3 need to be described and clarified. A role represents an abstract description of the system function. According to Gaia, a role is characterized mainly by permissions and responsibilities. Permissions represent access rights and limitations of information resources related to that role. Responsibilities, on the other hand, represent the functionality of the role. They are categorized into liveness and safety responsibilities, where liveness describes the normal execution cycle of an agent and is formalized by a well formed formula (wff) and safety describes invariants that an agent always maintains to avoid undesirable events from occurring and is represented in the form of predicates [11].

The role model of the crawler is depicted in Figure 4, where a short description of the crawler is given followed by a formalization of its permissions and its responsibilities. The figure illustrates under responsibilities section that the crawler either searches for new documents by querying a search engine or crawls through documents in its input queue. Documents resulting from searching are passed to the topic agent to be displayed and classified by the user. These documents are passed back to the crawler in the input queue to navigate through the containing links and enqueue the resulting documents in the output queue.

Role Schema: Crawler

Description:
This role involves collecting Web documents through searching for keywords and navigating through resulting documents in search for linked documents that seem relevant

Protocols and activities:
FormulateQuery, SendQueryToSearchEngine, GetNextSearchPage, ExtractLinks, ClassifyLink, EnqueueLinks, EnqueuePositive Links, DequeueURL, DownloadPage

Permissions:
Reads *Inpout Queue //containing documents to be crawled*
linkSet // dataset of link texts built during learning stage.

Changes *Output Queue // documents collected by the crawler*
Documents Folder // contains downloaded documents

Responsibilities:
Liveness:
Crawler = (Search I CrawlInputDocumets)^{w}

Search – FormulateQuery.SendQueryToSearchEngine. (GetNextSearchPage. Extract Links. EnqueueLinks)$^{+}$

CrawlInputDocuments=(DequeueURL.DownloadPage.ExtractLinks. ClassifyLinks. EnqueuePositiveLinks)$^{queue\_Size}$

Safety:
When linputQueue is Empty OR outputQueue is Full sleep

Figure 4: The role model of the Crawler.

With the linked structure of the Web, a Web page is typically linked to other pages that are completely out of the topic to the content of the containing pages. A typical page on CNN, for instance, reviewing the late American hurricane may be relevant to a topic agent searching for documents about "Natural Catastrophes" but may lead to pages of other current events like the world cup, which are totally irrelevant to the topic at hand. Therefore, focused crawling[1] is performed to reduce the number of irrelevant documents retrieved. Naïve Bayesian Classifier (NBC) is used to classify each link against a dataset of link texts – that is learnt by the DSReader during the training stage – before deciding to enqueue the document of that link in the output queue or not.

-- AGENT STATES

To fulfill its role, each agent goes through several states throughout its lifetime. A transition from a state to another may occur automatically upon a task completion or in response to an external stimulus from the surrounding environment. For example, an agent may move to a different state when an event is signaled, a message is received, or a signal is detected. Modeling the different states which an agent undergoes is very helpful to understanding an agent-based system.

Figure 5 depicts the top level state diagram of the TopicAgent. Whenever the system is started, the TopicAgent is in the idle state waiting for activation. The system activates topic agents belonging to the current user and each topic agent activates in its turn all agents under its supervision -namely the crawler, DSReader, and ProfileAgent. The TopicAgent then waits for an external request to be handled.

The TopicAgent receives messages and detects events triggered from the interface agent, the crawler, the dataset reader and the profile agent. When documents are ready in its output queue, the crawler sends a message notifying its topic agent. Figure 6 illustrates how the topic agent reacts to this message. It either moves to the displaying or transferring state based on whether a topic profile has been learnt or not. When displaying, it passes documents to the interface agent to be displayed to the end user. However, in case a profile has already been learnt, the topic agent sends documents to the dataset reader to be scanned and classified later by the profile agent.

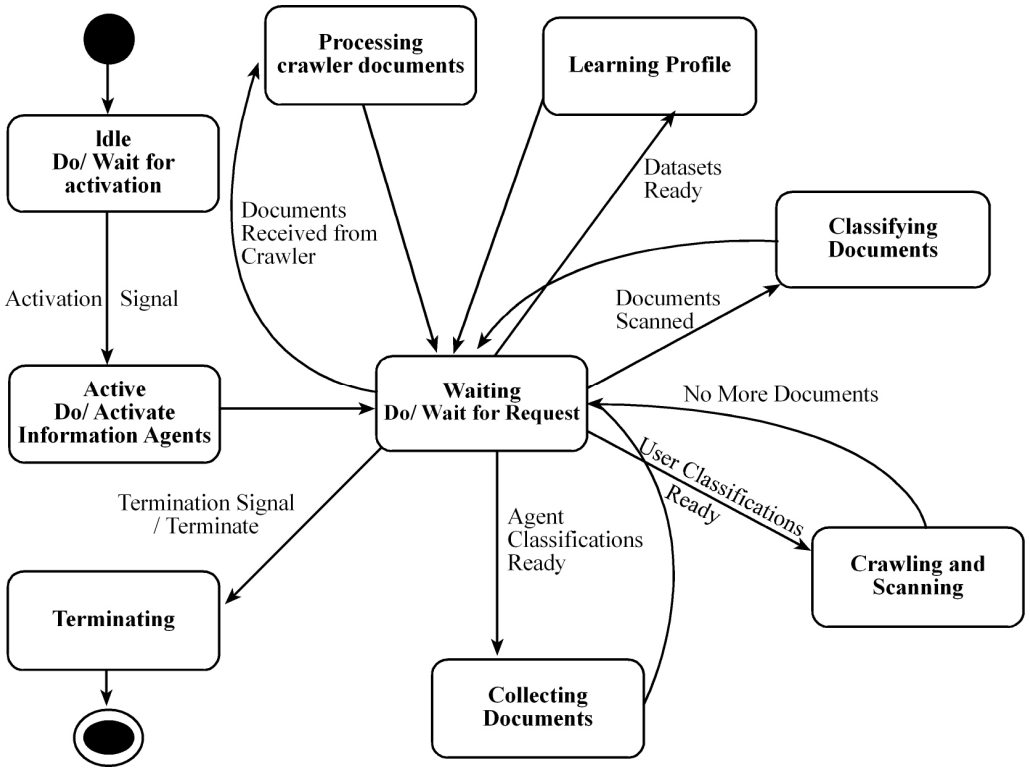

Figure 5: Top level state diagram of the topic agent (TopicAgent).

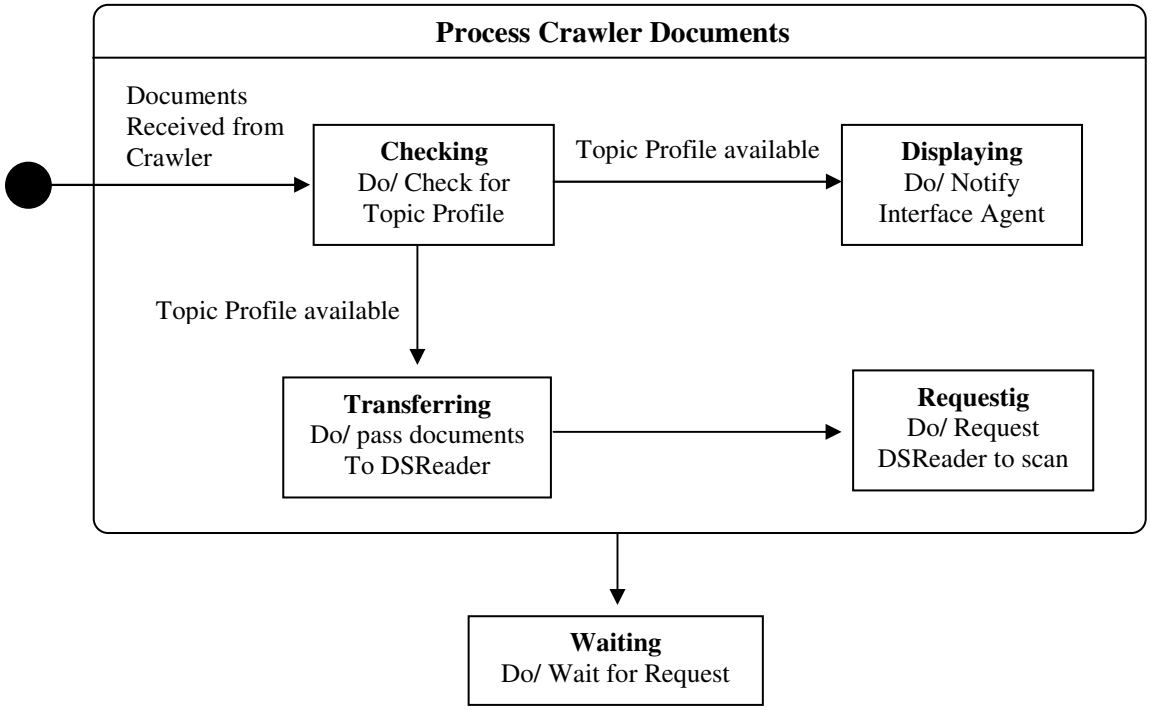

Figure 6: The "Process Crawler Documents" state.

--AGENT INTERACTIONS

Interactions among agents are captured in the analysis phase of Gaia by an interaction model and elaborated in the design phase by an acquaintance model. The former model is restricted to representing the nature and purpose of agent interactions and is abstracted away from any sequence of steps. Likewise, the latter is abstracted away from details related to the specific messages sent from agent to agent and is just concerned about representing the communication links between agents in the form of a graph whose nodes represent agents and arcs represent links between agents [11,12]. AUML provides a set of artifacts that represent dynamic relationships among agents where these relations are captured and detailed. One of these artifacts is called the Agent Interaction Protocol (AIP) diagram [1,7] and is equipped with abstractions tuned for agent communications with their messages and communication acts.

Figure 7 illustrates an AIP diagram showing the interaction between the crawler and the user agents. The diamond shape in the arrow indicates a decision accompanying a message where an ‘x’ inside represents an exclusive or. For example, the figure illustrates that upon receiving a request message from the user agent; the crawler either refuses or proposes documents.

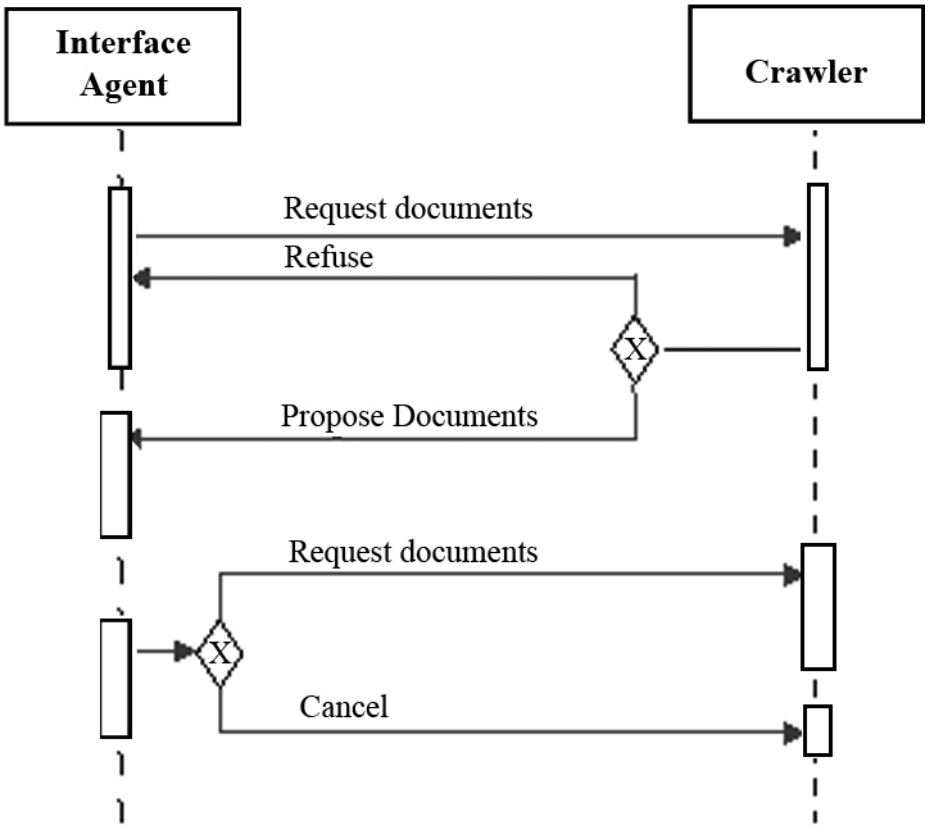

Figure 7: An Agent Interaction Protocol (AIP) diagram for the crawler and interface agents.

CONCLUSIONS

In this paper, FAIN, an information agent-based filtering system has been introduced. The design of the system has been illustrated using the Gaia methodology and the AUML notations. The proposed system interleaves browsing with crawling and searching for new documents. The resulting stream of documents is continuously evaluated according to a learned user profile that is constructed by training through examples and persistently tuned to better match his interests based on feedback. Not only does the system react to the dynamics of the environment represented in the subset of the located documents from the web and the changing user interests, but also it proactively takes the initiative in suggesting documents that may inspire potential interests of the user.

The illustrated case study has shown that the application of the AUML notation with the Gaia methodology provides a good support for guiding and covering the development process of agent-based systems. While Gaia seems to offer a powerful methodology that guides the designer throughout the development life cycle with sound concepts that go in harmony with the agent approach, AUML notation was found to complement Gaia by offering notations that aids greatly in elaborating on the artifacts that are provided by Gaia at the conceptual level.

ENDNOTE

1. "Focused Crawling" is concerned with gathering relevant Web pages on a particular topic.

REFERENCES

[1] Bauer, B., et al., "Agent UML: A Formalism for Specifying Multi-Agent Software Systems." *International Journal of Software Engineering and Knowledge Engineering*, Vol. 11, No.3, pp.1-24, 2001.

[2] Belkin, N. J. and Croft, W. B., "Information Filtering and Information Retrieval: Two Sides of the Same Coin?" *Communications of the ACM*, Vol. 35, No. 12, 1992.

[3] El-Khouly, M. M. and Badr, I. A., "Combining Learning with Evolution to Filter Search Engine Results." *International Symposium on Robotics and Automation (ISRA 2002)*, Toluca, Mexico, September 1-4, 2002.

[4] El-Khouly, M. M. and Badr, I. A., "Effectiveness of GA-NBC Using FAT." *WSEAS International Conference on E-Activities*, Singapore, December 9-12, 2002.

[5] Franklin, S. and Grasser, A., "Is It an Agent, or Just a Program?" *in Proceedings of the 3rd Conference about Agent Theories, Architectures, and Languages*, 1996.

[6] Kobayashi, M. and Takeda, K., "Information Retrieval on the Web", *ACM Computing Surveys*, Vol. 32, No. 2, 2000.

[7] Odell, J., et al., "Representing Agent Interaction Protocols in UML." in Ciancarini, P. and Wooldridge, M. (Eds.), *Agent-Oriented Software Enginering*, Springer-Verlag, Berlin, pp. 121-140, 2001.

[8] Perez, E., "Agents (One-Stop Shopping) for Researchers." *Online*, Vol. 26, No. 2, pp. 20-25, 2002.

[9] Perez, E., "Intelligent Agents: It's Nice to Get Stuff Done for You." *Online*, Vol. 26, No. 3, pp. 51-56, 2002.

[10] Sarhan, E. A., et al., "Combining Learning with Evolution in an Information Filtering Agent." *in Proceedings of the 8th International Conference on Artificial Intelligence Applications*, 2000.

[11] Wooldridge, M., et al., "A Methodology for Agent-Oriented Analysis and Design." *in Proceedings of the 3rd International Conference on Autonomous Agents*, Seattle, WA, pp. 69-76, 1999.

[12] Zambonelli, F., et al., "Agent-Oriented Software Engineering for Internet Applications." in Omicini, A., et al., (Eds.), *Coordination of Internet Agents,* Springer-Verlag, 2001.

[13] Zambonelli, F., et al., "Developing Multiagent Systems: the Gaia Methodology." *ACM Trans on Software Engineering and Methodology*, Vol. 12, No. 3, pp. 317-370, 2003.

[14] Zambonelli, F., et al., "Multi-Agent Systems as Computational Organizations: The Gaia Methodology." in Henderson-Sellers, B. and Giorgini, P. (Eds.), *Agent-Oriented Methodologies*, Idea Group, 2005.