



Interface Agent:

Papered by:

Mutaz Abdel-Razzaq Hassan Al-Mefleh

88254

Guide Line

1. Agent.
2. Classification of Intelligent Agents.
3. High-Level Classification of Agents.
4. General Properties of Agents.
5. General Model of an Agent Platform.
6. Types of Agents.
7. Definition of Interface Agents.
8. Classification of Interface Agents.
9. Example Interface Agent System.
10. Learning of Interface Agents – usually from the user.

Cont'()

11. Benefits of Interface Agents.
12. Criteria for Building Interface Agents.
13. Approaches to building Interface Agents.
14. The purpose of an Interface Agent.
15. Example of Interface Agent

Agent:

- Janson in (1997) gave the major ones where some of them are definitions and some are descriptions (description in terms of their tasks, autonomy, and communication capabilities).
 - Agents are semi-autonomous computer programs that intelligently assist the user with computer applications. Agents employ artificial intelligence techniques to assist users with daily computer tasks, such as reading electronic mail, maintaining a calendar, and filing information. Agents learn through example-based reasoning and are able to improve their performance over time.

Cont'()

- Agents are software robots. They can think and will act on behalf of a user to carry out tasks. Agents will help meet the growing need for more functional, flexible, and personal computing and telecommunications systems. Uses for intelligent agents include self-contained tasks, operating semi-autonomously, and communication between the user and systems resources. (Jansen, 1997)

Classification of Intelligent Agents

- Moukas and Maes (1998) proposed on distinguishes between three types of agents according to the very nature of their intelligence:
 - User programmed agents. Here, the user has to provide the "rules" to the agents so the agents follow the actions. These agents are considered to be "not very smart."
 - Artificial Intelligence engineered agents: These agents are considered to be smarter than the previous. They are based on traditional Artificial Intelligence Techniques.

Cont'()

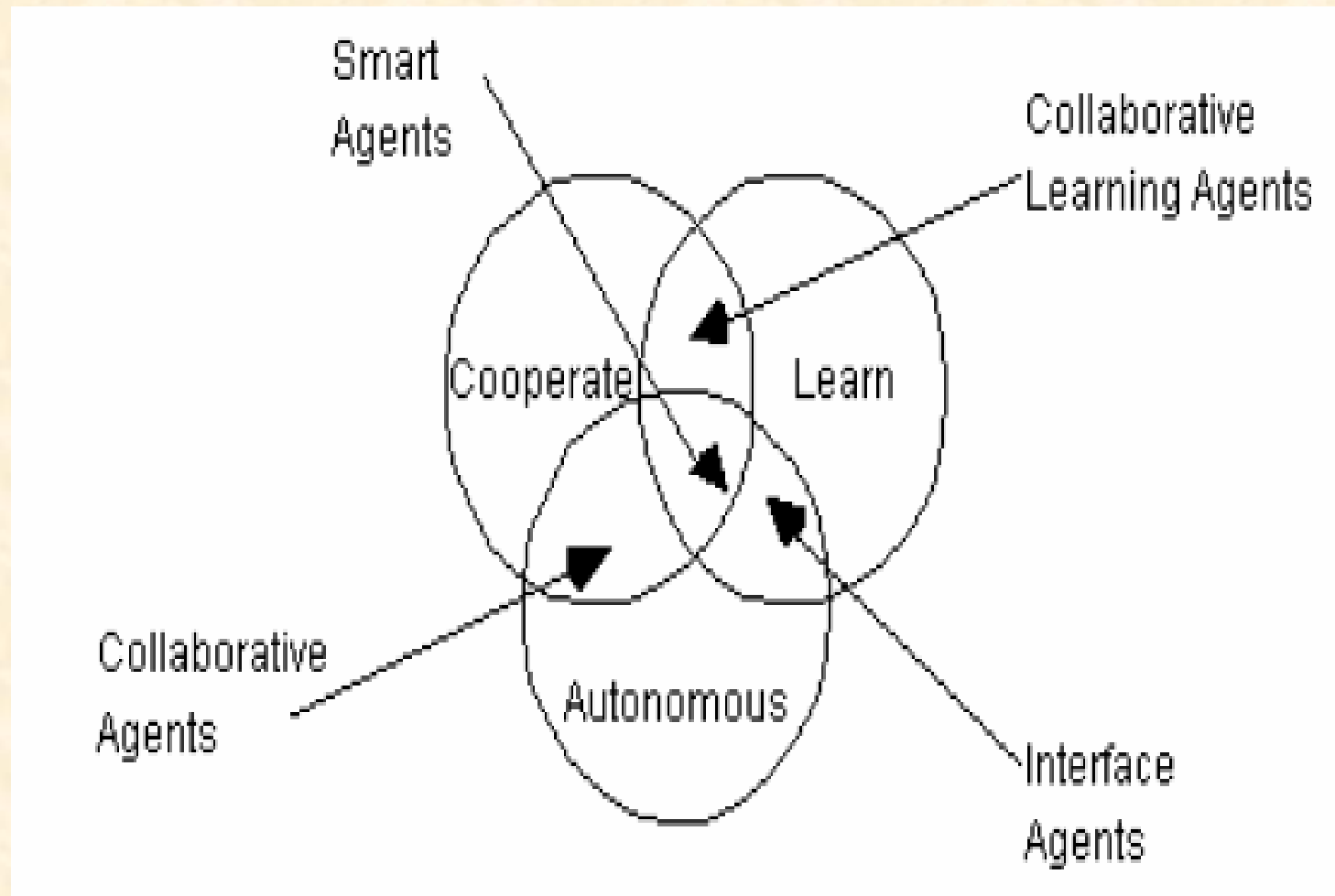
- Learning agents: These agents are distinguished that they "program themselves". They learn from their users, detect their user's actions and the behavior among users. Beside that, they can learn from other agents. We mean this category by the term "intelligent agents".
- Also, agents can be divided according to the type of tasks they perform for their users.
- The following table demonstrates the classification of agents based on their task which is proposed in www.BotSpot.com, see table 1:

Table 1:

classification of agents based on their task

#	Category	Sub-Categories
1	Search Bots	Image Bots, Newsgroup Bots, Meta-search Bots, On-line investigation, Music Bots
2	Shopping Bots	Auction Bots, Shopping Organizers, Shop Bots, Stock Bots
3	Tracking Bots	E-mail Notification Bots, News Bots, Spam Filtering Bots, Spy Bots, Weather Bots, Web Monitoring Bots
4	Artificial Life Bots	Adult Characters, Characters, Chatter bots, Personal Assistant Bots
5	Download Bots	Download Managers, File-sharing Bots, Off-line Browsing Bots
6	Web Development bots	Indexing Bots, Referencing Bots, Site Management Bots
7	Surf Bots	Accelerator Bots, Child Protection Bots, Form Filling Bots, Pop-up Killer Bots, Privacy Protection Bots, Spy ware Detection Bots
8	Games Bots	Build a Bot, Logic Bots, Pure Fun, Simulation Games

A High-Level Classification of Agents



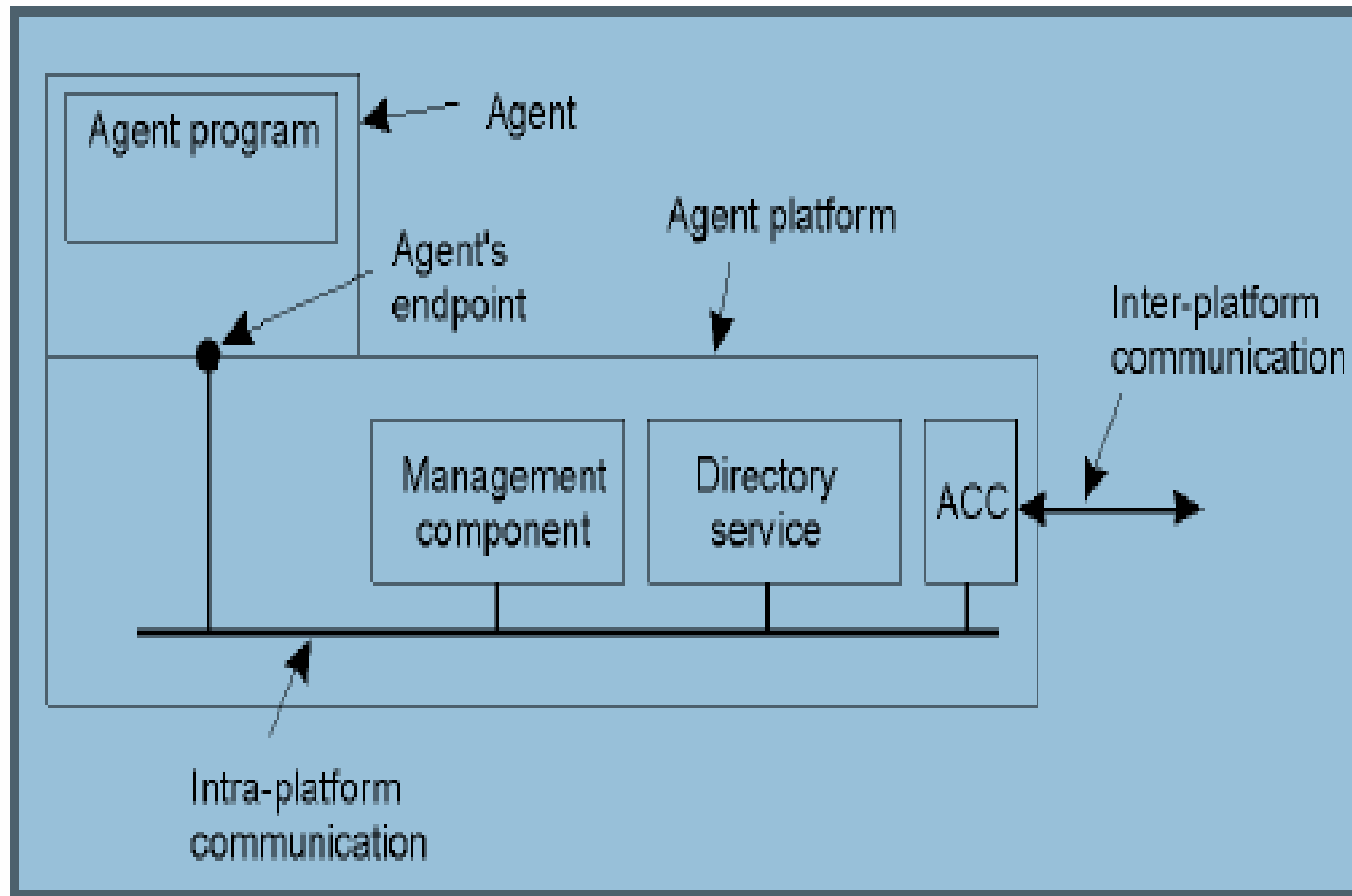
General Properties of Agents

Property	Common to all agents?	Description
Autonomous	Yes	Can act on its own
Reactive	Yes	Responds timely to changes in its environment
Proactive	Yes	Initiates actions that affects its environment
Communicative	Yes	Can exchange information with users and other agents
Continuous	No	Has a relatively long lifespan
Mobile	No	Can migrate from one site to another
Adaptive	No	Capable of learning

General Model of an Agent Platform

- Management component
 - Keeps track of the agents for the associated platform.
 - Creates and deletes agents.
 - Looks up the current end points of agents (naming service).
- Directory service
 - Look up what other agents on the platform have to offer
- Agent Communication
 - takes care of reliable and ordered point to point communication between agent platforms

Cont'()



Types of Agents

- Collaborative Agents
- Interface Agents
- Mobile Agents
- Information Agents
- Reactive Agents
- Heterogeneous Agent Systems
- Hybrid Agents (one or more of the above)
- Smart Agents – not a reality, dream of agent researchers

Definition of Interface Agents

- Moukas and Maes (1998) gave description of interface agents as follows: “Instead of user initiated interaction via commands and/or direct manipulation, the user is engaged in a co-operative process in which human and computer agents both initiate communication, monitor events and perform tasks.

Cont'()

- The metaphor used is that of a personal assistant who is collaborating with the user in the same work environment.”
- Moukas and Maes inferred from the description that the concept behind it is to let the user to delegate some tasks to an agent as assistance.
- The goal is to reduce the workloads of users by making personalized agents which handle some personal work delegated by the users.

Classification of Interface Agents

- Middleton (2001) classified user interface agents based on the techniques they use such as machine learning techniques or user modeling types.
- He gave 10 application domain based categories and then he compared them (agent by agent) to some specific taxonomy of interface agents as follows:

Cont'()

1. *Character-based agents*

- Deal with advanced “character” based interfaces, which represent real world characters

2. *Social agents*

- Talk to other agents to share information:
- Recommender systems: type of social agent. They are called "collaborative filters", they find relevant items based on others' recommendations.

3. *Agents that learn about the user*

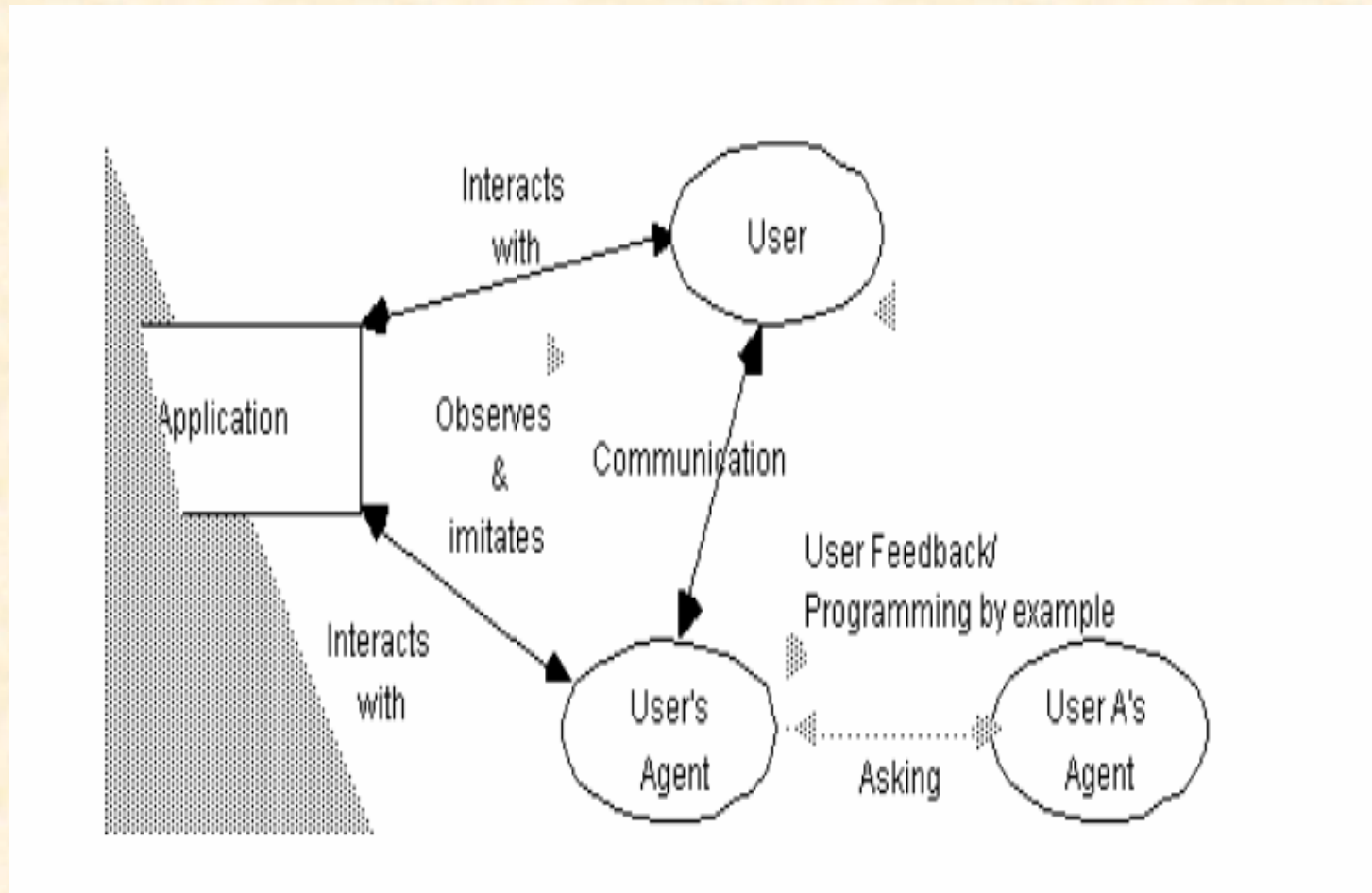
- Monitor user behavior
- Receive user feedback
 - Explicit feedback
 - Initial training set

4. *Programmed by user*

5. *Agents with user models*

- Behavioral model: the result of monitoring the user behavior while s/he is doing an activity.
- Knowledge-based model: the result of questionnaires and studies of users, arranged into a set of heuristics.
- Stereotypes: applied to both cases (Behavioral and Knowledge-based models), to classify the users into stereotypes (groups), in order to apply generalizations to people in those groups.

Example Interface Agent System



Learning of Interface Agents – usually from the user

- By observing and imitating the user
- Through receiving positive and negative feedback from the user
- By receiving explicit instructions from the user
- By asking other agents for advice (learning from peers)

Benefits of Interface Agents

- Reduce the work for the end user and application developer
- Can adapt to its user's preferences and habits
- Know-how among the different users in the community may be shared (learn from their peers)
- Collaborating with a user may not require an explicit ACL as one required when collaborating with other agents

Criteria for Building Interface Agents

- Competence
 - How does an agent acquire the knowledge it needs to decide when to help the user, what to help the user with and how to help the user?
- Trust
 - How can we guarantee the user feels comfortable delegating tasks to an agent?

Approaches to building Interface Agents

1. Rule-based Approach

- The rule-based approach features a collection of user-programmed rules for processing information related to a particular task. An end user program allows users to program rules for processing information for a task. However, this approach has several problems. The agent acquires its knowledge about how, when, and how much to help the user by extensive programming by the user, which negates the purpose of an agent as a tool which will save effort on the part of the user.

2. Knowledge Base Approach

- This interface agent has domain-specific background knowledge about the application and user to recognize plans and contribute to users' tasks. This approach has several problems as well. Where an end-user program approach requires a great deal of work on the part of the user, this approach requires a huge amount work for a knowledge engineer. The knowledge engineer must outfit an interface with large amounts of knowledge about the application, the domain and how the agent can help the user.

3. Machine Learning Approach

- This approach addresses problems encountered by the rule-based and knowledge-engineered approaches. This approach requires less initial work, and adapts over time. The agent acts as a personal assistant to cooperate with a user on a task, but makes allowances for user override. The agent learns by:
 1. Observing and imitating user
 2. Adapting based on user feedback
 3. Trained by user by example
 4. Ask for advice from other agents

Cont'()

- Machine learning uses memory-based reasoning combined with rules to model each user's habits. This approach achieves a level of personalization impossible previously available except through user intervention. However, these agents have their problems as well. Learning agents have a slow learning curve, requiring a sufficient number of examples before it can make accurate predictions. These agents also encounter problems when dealing with completely new situations. To address these problems, agents may learn from existing agents to get up to speed quickly. Over time, agents learn to be selective when learning from other agents, by learning to trust the suggestions of other agents more than others for various classes of situations.

The purpose of an Interface Agent is one of the following:

- Help the user during his work. For instance, the user repeats an action three times without obtaining the desired results (this may be known if the user undoes the action). The agent detects this and gives assistance in order to complete the action (Maes 98).
- Improve productivity. For instance, the user repeats the same sequence of actions several times (meanwhile the agent is watching him). On trying the sequence again, the agent raises its hand and solicits to perform the sequence by itself (Lieberman and Maulsby 1996).

Cont'()

Not as usual, but quite original, is the next one:

- Increase the functionality of an application. From a software engineering point of view, this is a very convenient solution, since in many cases the changes needed in an application to increase its functionality are greater than the ones required to add an agent. This is one of the objectives of our work.

Example of Interface Agent

1. *Auction/market Domain*

- **Kasbah** is a market system in which each user has an agent. The user programs the agent with a buying behaviors profile, and the agent negotiates to buy and sell items for the user. *Sardine* is an auction agent that tries to purchase an airline ticket for the user, based on some specified preferences. The user's agent negotiates with travel agents to secure the best deal.

2. *Believable/entertainment Domain*

- **ACT** is an addition to the ALIVE system. It is a creature within the ALIVE world, observing the user and learning chains of actions. It tries to help the user by completing new action chains in the pattern of previous ones. ALIVE is a “magic mirror” system to a 3D world. Interactive agents (such as a dog) exist for users to play with. Gesture recognition, and competing goal architecture is employed.

3. *Email Filtering Domain*

- **MailCat** filters email by providing a choice of folders to the user. TF-IDF vectors are created for existing emails, and cosine similarity used to match new emails. The user has the final say, choosing one of the suggested folders or moving messages manually. *MAGI* filters emails, monitoring user behavior and receiving relevance feedback. CN2 and IBPL are used to classify emails.

4. *Expert Assistance Domain*

- **Coach** is a LISP help system that monitors user mistakes and offers unsolicited advice. A knowledge-based user model is supported, with the concept of user experience stereotypically represented. Heuristics adjust the model based on user mistakes. *Eager* automates observed repetitive HyperCard actions. It monitors the user looking for behavior patterns, and creates helpful macros from them.

5. Matchmaking Domain

- **ExpertFinder** monitors users' Java code and finds people who use the same classes. TF-IDF vectors represent code files, and cosine similarity is used to find similar people. *ReferralWeb* builds a social network from publicly available web pages. People's names are extracted from pages, and co-occurrence of names within pages implies a social connection. Queries such as "list docs close to Mitchell" can thus be issued.

6. Meeting Schedulers

- **CAP** is a calendar manager, monitoring email and scheduling software to detect meeting patterns. Decision trees (ID3), using information gain to select features, are converted to production rules. *Meeting scheduling agent* schedules meetings by learning repetitive actions the user performs. Memory-based reasoning and reinforcements learning are used. Users can give explicit feedback.

7. *News Filtering Domain*

- **ANATAGONOMY** is based on the Krakatoa Chronicle, providing a personalized newspaper. Implicit feedback from user activity has been added. *Butterfly* finds interesting conversations within Usenet newsgroups. The user initially provides keywords, and term frequency similarity between newsgroups and the user's profile is computed.

8. *Recommender Systems*

- **GroupLens** recommends newsgroup articles based on other's ratings. Users are asked to rate every article they read, and Pearson correlation coefficient-based prediction is used to find similar users. *PHOAKS* monitors newsgroup articles, and extracts web link recommendations by a set of heuristics. A set of collective recommendations for topics is thus compiled.

9. *Web Domain*

- **CiteSeer** helps users perform keyword searches for publications by using citations in documents. Heuristics extract citations, titles and abstracts, then algorithms (TF-IDF, Likelt) classify publications based on stemmed words. Citation links lead to new search keywords. *ARACHNID* is a spider that crawls a digital library or the web, starting from the users' bookmarks, searching for user provided keywords. The user provides feedback on the pages found. A genetic algorithm is used in addition to reinforcement learning.

10. *Other domains*

- **CILA** is an agent tested in an artificial, abstract domain. It tests constructive induction-based learning against AQ15c and selective induction. User monitoring, relevance feedback, initial training sets and social collaboration with other agents are supported (in its abstract world). *CIMA* is a text prediction agent, which suggests completions of sentences in a text editor. Heuristics learn from observed examples, hints and partial specifications.

Reference

- Jansen, J., "Using an Intelligent Agent to Enhance Search Engine Performance.", First Monday 2, no. 3 (March 3, 1997).
http://www.firstmonday.dk/issues/issue2_3/jansen/index.html
- Moukas, A. and Maes, P. (1998). "Amalthea: An Evolving Multi- Agent Information Filtering and Discovery System for the WWW." Autonomous Agents and Multi-Agent Systems, 1(1): 59-88.
- S. E. Middleton, "Interface Agents: A Review of the Field", Technical Report Number ECSTR-IAM01-001, University of Southampton, August 2001
- <http://ksi.cpsc.ucalgary.ca/courses/547-95/thomsona/agents.html> (2 of 5)
[22/02/2008] ١١:٥٧:٤٣ م