Interactive Integration of Information Agents on the Web

Yasuhiko Kitamura[1], Teruhiro Yamada[2⋆], Takashi Kokubo[3⋆⋆], Yasuhiro Mawarimichi[1⋆⋆⋆], Taizo Yamamoto[2†], and Toru Ishida[3]

[1] Osaka City University, Osaka 558-8585, Japan
kitamura@kdel.info.eng.osaka-cu.ac.jp
http://www.kdel.info.eng.osaka-cu.ac.jp/~kitamura/
[2] Laboratories of Image Information Science and Technology, Japan
[3] Kyoto University, Kyoto 606-8501, Japan

Abstract. World Wide Web contains a vast amount of different information stored in a huge number of distributed Web sites. Search engines and information agents have been developed to facilitate efficient information retrieval tasks from the Web. By integrating multiple search engines and information agents as an interoperable system, we increase the value of each of them. In conventional collaborative systems, the integration process is designed by system designers and is concealed from the end users.
This paper proposes an interactive multiagent-based interface called Multiple Character-agent Interface (MCI) where animated character-agents interact with each other and with the user for assisting in information retrieval. By using the MCI, even a novice user can create a team of information agents and can self-customize the agents through the interactions with them. We here report the architecture of MCI and two prototype systems based on MCI, Venus and Mars, which is a cooperative multiagent system for information retrieval, and Recommendation Battlers, which is a competitive multiagent system for information recommendation.

1 Introduction

World Wide Web (WWW) provide a means for disseminating and sharing information on the Internet and has been widely used for doing business, education, research, advertisement and so on. The amount of information stored on the Web is increasing day by day, but the more the information is stored, the more difficult to find. Search engines are the most popular tools to find Web pages. A search engine returns a list of URLs responding to the query keyword(s) submitted by the user, but it often returns too many URLs, which include a fair number of unrelated ones, to be processed by a human user.

⋆ Presently with SANYO Electric, Tokyo 113-8434, Japan.
⋆⋆ Presently with NTT Docomo, Kanagawa 239-8536, Japan.
⋆⋆⋆ Presently with NTT Advanced Technology, Kanagaww 210-0007, Japan.
† Presently with NTT West, Osaka 530-6691, Japan.

A number of information agents have been developed to replace or reinforce search engines [11, 12]. For example, Letizia [14], WebWatcher [10], and WebMate [5] learn the preference or interest of user by monitoring the history of Web browsing or searching performed by the user, and recommend suitable Web pages for the user or refine search keywords. Ahoy! [19] analyzes and filters the output of general-purpose search engine and returns a domain-specific output such as individual's homepages. Fab [2] is an information recommendation system that incorporates the collaborative filtering method.

Combining or integrating search engines and/or information agents adds more value to each system. Meta-search engines such as MetaCrawler [18] and SavvySearch [8] integrate the output of multiple search engines and succeed to improve the performance. BIG [13] is an information agent that intelligently and efficiently gathers information from multiple sources considering the trade-off between the quality of information and the constraints like time and cost. RETSINA [20] consists of three types of reusable agents; interface agents, task agents, and resource agents. An interface agent interacts with the user to receive a query from the user and returns results. A task agent solves domain-specific tasks through exchanging information with other agents. A resource agent provides access to a heterogeneous information source. Other multi-agent based systems, such as federated system [7], InfoSleuth [3], and LARKS[12], incorporate information brokering or information matchmaking mechanism.

In conventional collaborative systems such as mentioned above, the way to coordinate information agents or information resources is specified by the system designers and concealed from the users. Hence, the users are just allowed to submit a query to a fixed interface and to receive results from the system, but not allowed to change the combination of information agents nor the collaboration mechanism.

For example, RETSINA agents are reusable and their interface is open to the system designers but not to the user, so the user can just get access to the system only through the interface agent. In the federated system, the process of coordination among agents is specified in the ACL (Agent Communication Language) such as KQML [6]. The ACL provides an open interface to information agents but not to human users because it is difficult for a human user to communicate directly with agents using the ACL. Hence, neither system is designed to provide an open interface to the end user.

Each user has different demands or preferences for information retrieval. Some user may like some search engine and others may not. Hence, rather than just a ready-made collaborative system, we would like to have a framework where we can easily make a team of favorite information agents that work together and customize them flexibly.

This paper proposes the Multiple Character-agent Interface (MCI) as shown in Fig. 1 where multiple character agents, each of which represents an information agent that resides at a server, collaborate with each other on a client machine. We can view a character agent as the head of information agent whose body resides in a server. The collaborative actions for retrieving information are performed

by character agents on the client machine and displayed to the user. The user can get direct access to the agents by clicking and talking, so he/she can submit the request to each agent and can customize the agent's actions. The user can also make a favorite team of agents by calling them out to the client machine.

The advantages of MCI are as follows. The collaboration process among information agents is open to the user. The user can understand how the process goes and what happens in the system including some erroneous or inappropriate actions caused by an agent. In a conventional collaborative system, the user may not be able to notice such erroneous actions because the collaboration process is concealed from the user. In the MCI, the user can not only notice the error, but also fix it by tuning the corresponding agent or by replacing it with other appropriate agent. Hence, the user can customize the team of information agents through visible interactions with them.

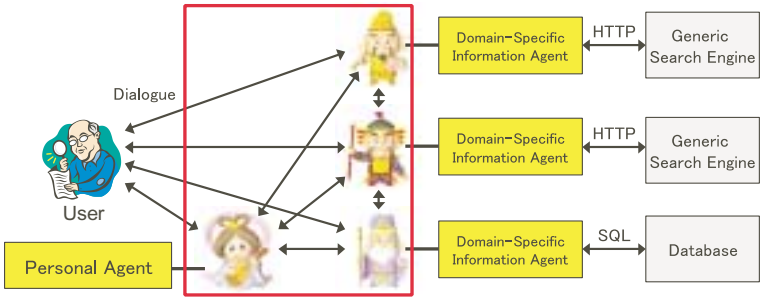

Fig. 1. Overview of Multiple Character-agent Interface.

Andre and Rist propose a similar system employing multiple character-agents [1], but their work mainly emphasize the advantage of multiple characters as a presentation media. Agents in their system reside and perform in a client machine whereas, in our system, agent's bodies reside independent servers in a distributed manner and agent's heads interact in a client machine.

Section 2 addresses the architecture and the implementation issues of the Multiple Character-agent Interface. Then, two prototypes that use MCI are introduced. Section 3 describes Venus and Mars which is a cooperative multi-agent system consisting domain-specific information agents and a personal agent. In Section 4, we introduce Recommendation Battlers, a competitive multi-agent system in which two character-agents competitively recommend restaurant information to the user.

2 Multiple Character-agents Interface

We show the system architecture of Multiple Character-agent Interface in Fig. 2. MCI consists of multiple character-agents; the body of each agent resides in a

different server. As an autonomous agent mentioned in [16], each agent recognizes actions taken by the user or other agents through data from sensors, interprets the actions, and responds through its actuator. Agent behavior is controlled by a program written in Q [9], which is an interaction design language being developed by Kyoto University for linking autonomous agents and humans.

Table 2 shows major sensor and actuator commands of a typical MCI agent. Other optional commands are available according to the functionality of agent.

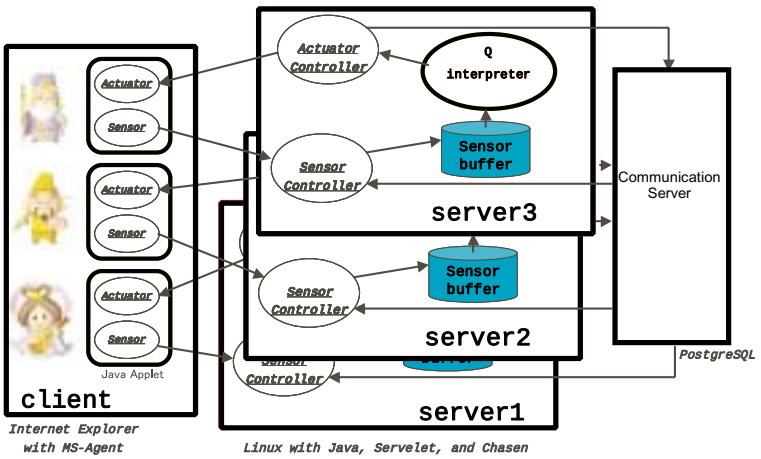

Fig. 2. Architecture of Multiple Character-agents Interface.

Here is an example of the behavior of an MCI agent as specified in Q.

```
((?feel)
   (!speak "Hello.")
   (!display_dialogue_box "prompt" :title "May I help you?"))
```

This rule specifies that the agent speaks "Hello." and shows a dialogue box with "May I help you?" when it is clicked.

```
((?find "time" :hour $hour)
   (if (and (>= hour 18) (< hour 6))
      (!speak "Good Night!"))
   (if (and (>= hour 6) (< hour 12))
      (!speak "Good Morning!"))
   (if (and (>= hour 12) (< hour 18))
      (!speak "Good Afternoon!")))
```

Sensor Commands:	
`(?feel)`	The agent checks whether it is clicked.
`(?hear $utterance [:from agent])`	The agent hears an utterance from other agents or its user.
`(?find $database_name :parameter parameter)`	The agent get information from the specified database.
Action Commands:	
`(!show agent)`	The agent appears.
`(!hide agent)`	The agent disappears.
`(!speak utterance [:to agent])`	The agent utters `utterance` to other agent.
`(!play_animation action)`	The agent performs the animated `action`.
`(!point direction)`	The agent points to the specified `direction`.
`(!display_dialogue_box type [:title title])`	The agent shows an dialogue box.
`(!present url)`	The agent shows a Web page at `url` on its browser.
`(!search keyword)`	The agent submits `keyword` to a search engine.

Table 1. Major sensor and actuator commands of MCI agent.

This rule specifies as the agent checks the time and then speaks the greeting words, "Good Morning!," "Good Afternoon!," or "Good Night!" according to the time.

To make an agent perform a complex behavior, rules are grouped in scenes as follows.

```
(scene1                          // Scene 1
  ((otherwise)
    (!show)                      // The agent appears
    (go scene2)))                // and goes to Scene 2.
(scene2                          // Scene 2
  ((?hear "Hello!")              // If the agent hears "Hello,"
    (!speak "Hello!"))           // he says "Hello."
  ((?feel)                       // If the agent is clicked,
    (!speak "I am itchy.")       // he says "I am itchy."
    (go scene3)))                // and goes to Scene 3.
(scene3                          // Scene 3
  ((?feel)                       // If the agent is clicked,
    (!speak "Please stop it.")   // he says "Please stop it."
    (go scene4)))                // and goes to Scene 4.
```

At the initial scene (Scene 1) the agent appears and moves to Scene 2. At Scene 2, he says "Hello." if he hears "Hello." If he is clicked, he says "I am itchy." and moves to Scene 3. At Scene 3, as the action is different from the one at Scene 2, he says "Please stop it." if he is clicked.

Information agents, which reside in distributed servers, are integrated through the integration of their characters (heads) on a client machine. Once an agent is

called to the client machine, it begins to interact with the user and other agents in a plug-and-play fashion.

Character-agents are implemented on the MS-Agent platform and controlled through Java Applets and Java Scripts running within Internet Explorer. By using multiple frames as shown in Fig. 3, we can realize multiple characters that interact with each other and the user. The MCI consists of a control frame and multiple agent frames. When the system is initiated, agent manager, user manager, and dialogue manager are downloaded in the control frame. Agent manager manages agent frames when agents are invoked or stopped. User manager manages users by using the Cookie mechanism. Dialogue manager controls the order of utterances in order not to make multiple agents talk at the same time. For a request of showing an agent, the agent manager downloads character controller, command receiver and transmitter from the agent server in an agent frame. Character controller is written in Java Script and controls a character agent. This corresponds to sensor and actuator modules in Fig. 2. Command receiver and transmitter are Java Applets connecting the character (head) and the server (body).

The user's actions (clicks or utterances) to an agent are sensed through its sensor applet (character controller) and forwarded to the sensor controller in the agent's server through the command transmitter. Actions taken by other agents are sensed through the communication server and the command receiver. Action commands from the agent's server is forwarded to the character through the command receiver and the character controller.

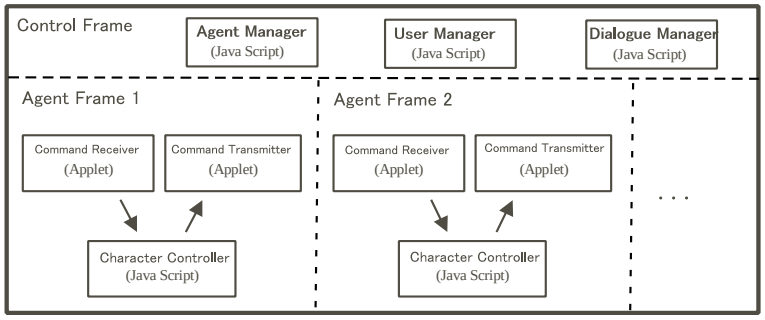

Fig. 3. Implementation of Multiple Character-agents Interface.

By adopting the above multiagent architecture, we can easily add or remove agents on the client machine. This allows a wide a variety of collaborative tasks for information retrieval to be achieved by making various teams of agents. Moreover, the failure of one agent does not affect the other agents seriously, so the performance of the whole team degrades only locally. While the agents should be autonomous to a large extent, some actions should be tightly controlled. To

this end we implemented the dialogue manager that prevents them from speaking at the same time.

3 Venus and Mars: A Cooperative Multiagent System for Information Search

Search engines are the most widely used tools for retrieving information from the Web. A drawback of conventional search engine is that it tends to return too many URLs with low precision as the amount of information increases on the Web. One approach to deal with this drawback is to build domain-specific search engines that can deal with some specific topics with high precision [19]. Furthermore, a platform where such domain-specific search engines or agents collaborate with each other looks promising.

Venus and Mars (VandM) [22] is an information retrieval system in which multiple character-agents work cooperatively to assist the user. As shown in Fig. 4, VandM consists of three types of character-agents. Kon-san is the information agent that locates Web pages about cooking recipe. When the user submits a Japanese query that include keywords about recipe, such as "I would like to eat a pork dish," Kon-san extracts one or more keywords about recipe ("pork") and submit the keyword(s), with keyword spices[1] for the recipe domain, to a general-purpose search engine[2]. It then receives search results from the search engine and shows them to the user through the Web browser. In case it receives too many results, it automatically asks for additional keywords to reduce the number of results.

Cho-san has knowledge about cooking ingredients and health. Responding to an utterance that includes keywords related to cooking ingredients or health, it utters comments about the relations between cooking ingredients and health, such as "Leeks are good for colds."

Pekko is the personal agent; it initially appears on the client machine and calls other agents. It chats with the user, and monitors the user's search history. When needed, it suggests some search keywords to Kon-san on behalf of its user referring to the user's history.

A snapshot of Venus and Mars is shown in Fig. 5. A typical cooking recipe search is given below.

(1)Pekko: "May I help you?"
(2)User: "I would like to eat a pork dish."
(3)Kon-san "OK. Let me locate some recipes that use pork. How do you like this?" (He shows a recipe page on the browser.)

[1] Keyword spices are domain-specific keywords to improve the performance of general-purpose search engine. [15] discusses how to discover keyword spice semi-automatically by using a machine learning technique. For example, tablespoon $\vee$ ($\neg$tablespoon $\wedge$ recipe $\wedge$ $\neg$home $\wedge$ $\neg$top) $\vee$ ($\neg$tablespoon $\wedge$ $\neg$recipe $\wedge$ pepper $\wedge$ $\neg$pan) is discovered for the recipe domain.

[2] http://www.goo.ne.jp/



Nickname	Pekko	Kon-san	Cho-san
Type	Personal Agent	Information Agent	Information Agent
Function	Chatting with the user. Learning the user's preference.	Searching WWW pages concerning cooking recipes.	Providing comments about combination of cooking ingredients and health.
Knowledge	User's profile	Keywords about recipes, ingredients, and seasoning.	Database about cooking ingredients and health.

Fig. 4. Character-agents in Venus&Mars.

(4)Cho-san "Pork is good for the digestion. It is a good way to get vitamin B."
(5)Kon-san "The number of search results is over 1000. Let me reduce the number. What kind of seasoning do you like?"
(6)Pekko: "I know Kitamura-san likes Japanese food."
(7)Kon-san: "OK. Let me find some Japanese recipes that use pork. How do you like this?" (He shows another recipe page on the browser.)
(8)Pekko: (Responding to a click from the user) "May I help you?"
(9)User: "I want a recipe that is good for recovering from a cold."
(10)Kon-san: "?"
(11)Cho-san: "Leeks are good for colds."
(12)Kon-san: "OK. Let me locate recipes that use leeks. How do you like this?" (He shows a recipe page that mentions leeks on the browser.)

In VandM, agents collaborate with each other in two ways. In the above dialogue steps, (5) to (7), Pekko assists Kon-san in reducing the number of search results. In utterance (5), Kon-san asks for a tip on seasoning, Pekko answers "I know Kitamura-san likes Japanese food." on behalf of the user by referring to his interaction history. Of course, if the user does not like Pekko's suggestion, he/she can correct Pekko's utterance by indicating his true preference directly to Kon-san through the dialogue box. Pekko recognizes this correction and updates the user's preferences stored in its database.

In the above dialogue steps, (9) to (12), Cho-san assists Kon-san. In this case, Kon-san cannot answer the request "I want a recipe that is good for recovering from a cold." because it has no knowledge about health. On the other hand, Cho-san has knowledge about cooking ingredients and health, so it makes the comment "Leeks are good for colds." Kon-san takes the comment as a clue to initiate a new search with the keyword of leek. This type of collaboration shows

a potential of VandM for realizing various type of information search by adding agents to the team. For example, if we add a restaurant recommendation agent to the team, the user's request for a recipe with salmon may also result in a recommendation of local restaurants specializing in salmon.

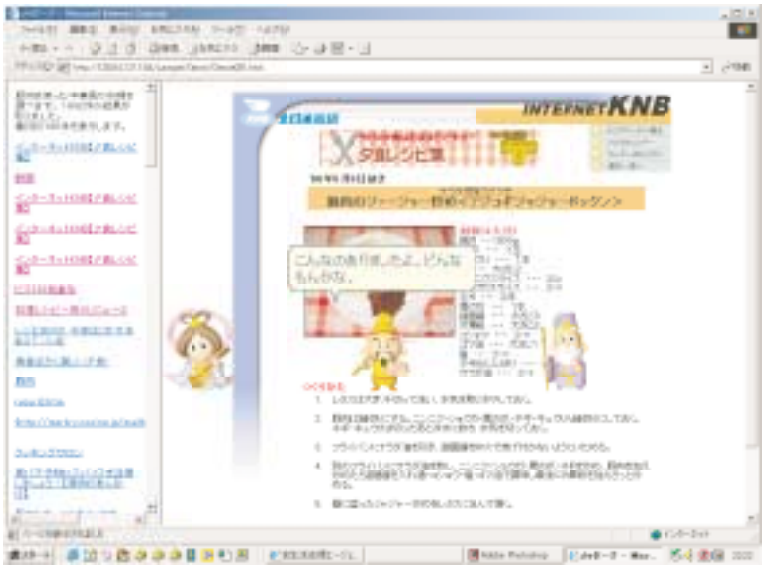

Fig. 5. A snapshot of Venus&Mars.

4 Recommendation Battlers: A Competitive Multiagent System for Information Recommendation

With the growth of e-commerce market, a large number of shopping sites are open on the Internet. Recently comparison shopping sites, such as www.dealtime.com, are also available. Such a site collects information from multiple shopping sites and provides a list of commodities sorted in order of the price. Though it is a very convenient tool to find the cheapest price, it does not provide direct interactions between buyers and sellers. We here propose a framework where an information user (buyer) and providers (sellers) directly interact with each other through information agents. This scheme gives a new style of e-commerce where multiple sales agents directly negotiate with the customer.

Recommendation Battlers(RB)[17] is a multiagent system for information recommendation in which two character-agents competitively recommend information. It consists of two restaurant recommendation agents; Peedy and Genie. Responding to a request from the user, Peedy collects relevant restaurant pages

from Yahoo! Gourmet[3] and extracts their addresses and average cost. It then calculate the distance from the nearest station by referring to the addresses. Genie also do the same process but it uses Kansai Walker[4]. The user and agents interact on a client machine as shown in Fig. 6. A typical dialogue is given below.

(1)User: "I would like to find a good noodle restaurant near Shinsaibashi station."
(2)Genie: (After collecting information from the Kansai Walker) "Do you like Restaurant A? The average cost is around 2,000 Yen and the distance from Shinsaibashi station is 90m."
(3)User: "I like it."
(4)Peedy: (After collecting information from the Yahoo! Gourmet) "Do you like Restaurant B? The average cost is around 2,500 Yen and the distance from Shinsaibashi station is 80m."
(5)User: "No, I don't like it."
(6)Peedy: "Do you like Restaurant C? The average cost is around 2,200 Yen and the distance from Shinsaibashi station is 80m."
(7)User: "Good."
(8)Genie: "I sure that you like Restaurant D. The average cost is around 1500 Yen and the distance from Shinsaibashi station is 70m."
(9)Peedy: "I have no more restaurant to recommend."

Fig. 6. A snapshot of Recommendation Battlers

[3] http://gourmet.yahoo.co.jp/
[4] http://www.walkerplus.com/kansai/gourmet/

In contrast to VandM, where agents cooperate, agents in RB compete with each other. Each monitors the other agent's proposals and the user's responses. It then proposes a new item that is more appropriate given the course of dialogue estimating the preference of user by using a rational proposal mechanism [17].

For example, in utterance (2), Genie proposes Restaurant A with the average cost of 2,000 yen and the distance of 90m. After the user accepts the proposal, the opponent Peedy proposes Restaurant B which looks better from his perspective because the distance is nearer than that of Restaurant A. However, the user does not accept the proposal, so Peedy proposes another Restaurant C. Through interactions with the user, Genie and Peedy continue to propose restaurants estimating the preference of user until one of them has nothing to propose.

5 Future Study and Conclusion

In this paper, we proposed the MCI where the user performs information retrieval tasks interacting with multiple information agents, and introduced two prototypes, Venus and Mars and Recommendation Battlers, using the MCI. Collaborative tasks performed in these prototypes are rather simple and a lot of future work still remains to build more advanced collaborative information agents.

Capability for collaboration The MCI agents start to collaborate with each other once they are called on the client machine in a plug-and-play fashion. Collaborative actions performed by agents in our current prototypes are simple such as association or extension of search keywords. To realize more advanced collaborative features, we need to incorporate collaborative mechanisms using coordination and negotiation techniques, which have been studied in the field of multi-agent systems [21], into our systems.

Capability for presentation In our current prototypes, collaboration among agents is presented as a conversational activity. Conversation among agents performed on a computer display is volatile and it may not be suitable for presenting a complex collaborative activity. Hence, we need to improve the presentation skill, for example, by creating a virtual space where agents can interact with not only other agents but also virtual objects. Such skills have much concern with the work being done by Andre and her colleagues [1].

Capability for interaction In our current prototypes, agents interact with the user mainly by using natural language, but just use a simple technique such as keyword extraction from an utterance in Japanese. Natural language is the most natural way of communication for human users, especially for novice users such as children and old people. For more complex interactions with the user, more advanced features of natural language processing are needed to be applied to our agents.

Current Internet-based information systems depend heavily on the Web technology. Since we can put not only text, but also image and audio, on a Web

page, the Web is highly expressive. The XML technology will further enhance the value of current Web system. On the other hand, the Web information system looks static because it just provide information in a page by page way. Character agents enhance the Web technology in another way and make it look more dynamic and interactive. To this end, a number of works have been done in the academic field [4] and some are commercially used such as Extempo [5], Haptek [6], Virtual Personalities [7], Artificial Life [8] and so on. The MCI provides a framework where multiple character agents are integrated as a collaborative system. This scheme may lead to a new generation of agent-based information integration systems after the XML age.

Acknowledgement

This paper reports the progress of a joint project of Kyoto University, Osaka City University, SANYO Electric, NTT West, and NTT Comware at LIST (Laboratories of Image Information Science and Technology) subsidized by NEDO (New Energy and Industrial Technology Development Organization). I thank Masahiko Kyosaka, Kiyomasa Okumura, Yuji Tomono, Hiroshi Morimoto, Mutsumi Ikeda, and Toshiki Sakamoto for their contribution to the development of the system.

References

1. Andre, E., Rist, T.: Adding Life-Like Synthetic Characters to the Web. Cooperative Information Agents IV, Lecture Notes in Artificial Intelligence 1860. Springer (2000) 1–13
2. Balabanovic, M., Shoham, Y.: Fab: Content-Based, Collaborative Recommendation. Communications of the ACM, **40**(3) (1997) 66–72
3. Bayardo, R.J., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewics, M., Shea, R., Unnikrishnan, C., Unruh, A., Woelk, D.: InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments. In Proceedings ACM SIGMOD International Conference on Management of Data (1997) 195–206
4. Cassell, J., Sullivan, J., Prevost, S., Churchill, E. (eds.): Embodied Conversational Agents. The MIT Press (2000)
5. Chen, L., Sycara, K.: Web Mate: A Personal Agent for Browsing and Searching. In Proceedings of the Second International Conference on Autonomous Agents (1998) 132–139
6. Finin, T., Labrou, Y., Mayfield, J.: KQML as an Agent Communication Language. In Software Agents, AAAI Press (1997) 291–316
7. Genesereth M.R.: An Agent-Based Framework for Interoperability. In Software Agents, AAAI Press (1997) 317-345

[5] http://www.extempo.com
[6] http://www.haptek.com
[7] http://www.vperson.com
[8] http://www.artificial-life.com

8. Howe, A.E., Dreilinger, D.: Savvy Search: A Metasearch Engine That Learns Which Search Engines to Query. AI Magazine, **18**(2) (1997) 19–25
9. Ishida, T.: Interaction Design Language Q: The Initial Proposal. In Proceedings 15th Annual Conference of Japanese Society for Artificial Intelligence, 2A2-03 (2001)
10. Joachims, T., Freitag, D., Mitchell, T.: WebWatcher: A Tour Guide for the World Wide Web. In Proceeding of the 15th Joint Conference on Artificial Intelligence (1997) 770–775
11. Klusch, M.: Intelligent Information Agents, Springer (1999)
12. Klusch, M.: Information Agent Technology for the Internet: A Survey. Journal on Data and Knowledge Engineering, **36**(3) (2001)
13. Lesser, V., Horling, B., Klassner, F., Raja, A., Wagner, T., Zhang, S.X.: BIG: An agent for resource-bounded information gathering and decision making. Artificial Intelligence, **118**(1-2) (2000) 197–244
14. Lieberman, H.: Letizia: An Agent That Assists Web Browsing. In Proceedings of the 14th International Joint Conference on Artificial Intelligence (1995) 924–929
15. Oyama, S., Kokubo, T., Yamada, T., Kitamura, Y., Ishida, T.: Keyword Spices: A New Method for Building Domain-Specific Web Search Engines. In Proceedings 17th International Joint Conference on Artificial Intelligence (2001) (in press)
16. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall(1995)
17. Sakamoto, T., Mawarimichi, Y., Kitamura, Y., Tatsumi, S.: Competitive Information Recommendation System Using Multi-Characters Web Information. In Proceedings 15th Annual Conference of Japanese Society for Artificial Intelligence, 1F1-01 (2001)
18. Selberg, E., Etzioni, O.: Multi-Service Search and Comparison Using the MetaCrawler. In Proceedings of the 4th International World Wide Web Conference (1995) 195–208
19. Shakes, J., Langheinrich, M., Etzioni, O.: Dynamic Reference Sifting: A Case Study in the Homepage Domain. In Proceedings of Sixth International World Wide Web Conference (1997)
20. Sycara, K., Zeng, D.: Coordination of Multiple Intelligent Software Agents. International Journal of Cooperative Information Systems **5**(2&3) (1996) 181–211
21. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. The MIT Press (1999).
22. Yamada, T., Kokubo, T., Kitamura, Y.: Web Information Integration Using Multiple Character Interface. In Proceedings 15th Annual Conference of Japanese Society for Artificial Intelligence, 1F1-05 (2001)