A Taxonomy of Web Agents

Zhisheng Huang*, Anton Eliëns*,+, Alex van Ballegooij+, and Paul de Bra**,+

*Free University of Amsterdam
Department of Mathematics and Computer Science
De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands
{huang, eliens}@cs.vu.nl

+Center for Mathematics and Computer Science(CWI)
P.O.Box 94079, 1090 GB Amsterdam, The Netherlands
{eliens,alexb}@cwi.nl

**Eindhoven University of Technology
Department of Mathematics and Computing Science
Postbus 513, 5600 MB Eindhoven, The Netherlands
debra@win.tue.nl

Abstract

In this paper we will propose a taxonomy of web agents, which encompasses agents that provide a text-based interface to for example information retrieval services as well as avatar-embodied guides that help visitors to navigate in virtual environments. Our taxonomy must be regarded as an instrument to delineate targets for research and the realization of prototype applications that demonstrate the usefulness of agent-based intelligence on the Web. In addition, we deploy our agent-taxonomy to establish the implications particular target applications have with respect to software architecture and computational resources.

Key words: web agent, taxonomy, virtual community

1 Introduction

There is a lot of interest and work in the research and development of agent technology with applications on the Web. Many types of web agents have been proposed in recent years, which range from domain-dependent agents, like e-commerce agents, information gathering agents, expertise seeking agents, travel assistant agents, to function-dependent agents, like negotiation agents, co-operating agents, problem solving agents. In addition, we like to mention the existence of virtual community agents. The natural questions related to that phenomenon are: what are the relations among so many different types of web agents? Are they redundant, or overlapped? Is there any taxonomy to classify them?

In this paper, we will propose a taxonomy of web agents, which encompasses agents that provide a text-based interface to, for example, information retrieval services, as well as avatar-embodied guides that help visitors to navigate in virtual environments. Our taxonomy must be regarded as an instrument to delineate targets for research and the realization of prototype applications that demonstrate the usefulness of agent-based intelligence on the Web. In addition, we deploy our agent-taxonomy to establish the implications that particular target applications have on the software architecture and computational resources. This research is part of a Dutch research project WASP, for Web Agent Support Program. As part of this research project we are developing PAMELA, a Personal Assistant for Maintaining Electronic Archives. The taxonomy we present in this paper is meant to serve in selecting a suitable "type" of agent architecture to use in the WASP project.

This paper is organized as follows: Section 2 gives a brief overview of intelligent services on the Web. Section 3 discusses what role agents can play in virtual environments. As a remark, our interest in virtual environment justifies why we consider the dimension "2D versus 3D" in Section 5. Section 4 elaborates on the existence of agents in computer games, which can be considered as another application of virtual community agents. In Section 5, we consider several dimensions of web agents, propose the taxonomy, and discuss types of web agents, their relationship, and the possible applications. Section 6 indicates research issues which need to be addressed to realize the potential of intelligent web-agents in virtual environments. Section 7 concludes the paper.

2 Intelligent services on the web – an overview

In a relatively short time, the Web has become a de facto standard for the dissemination and retrieval of information. Due to the exponential growth of the Web and the information it provides, finding relevant information has become more and more difficult. In particular, browsing is in most cases no longer appropriate for the user searching for specific information. It is our view that, in the near future, access to the Web will increasingly be mediated by intelligent helper applications, software agents, that assist the user in finding relevant and interesting information. As testified by a large body of literature, including [6, 7, 17, 15], this view is shared by many authors and developers.

During the first years of the Web it was feasible to create a fairly complete subject index of the whole World Wide Web. Yahoo was probably the most prominent one. Besides such subject indexes there were search engines that provided full text search on virtually the whole Web. Neither approach has proved sufficient. In order to better serve users Web "portals" have emerged, combining searching and browsing, and most indexes and search engines now offer this combined access to information. Moreover, several sites are beginning to offer *personalized* portals that can be customized by the user and that may even adapt to the user's interests automatically. It is this personalization that is the step that moves portals from being general information services to towards personal agents that point users towards relevant information and that hide stuff that is outside the user's field of interest.

A similar evolution exists in other areas like e-commerce and education. Personalized agents offer buying suggestions. Some can even automatically buy products at the best price or other conditions. Adaptive learning systems help users in studying a subject of a course by showing which pages to read in which order in order to reach that objective. More about adaptive systems can be found in [5].

As a working definition of the notion of *agents*, we adopt the characterization given in [14], which includes characteristics such as *responsiveness*, *pro-activity* and *socialness*. These characteristics, obviously, are strongly metaphorical, and are meant to stress the need for flexible, adaptive approaches to provide assistance to the user.

User assistance may involve intelligent desktop support (such as the infamous paper clip), information filtering and storage (to augment search engines), the delegation of tasks (such as responding to email, or setting a shared agenda), and the maintenance of user preferences and profiles (for example to make a proper selection of musical material).

Clearly all these applications benefit from a metaphor that stresses some kind of situated intelligence, that is software processes that can interact with other such processes (and humans) and that may be adapted to the individual users' needs and preferences.

Avoiding a discussion of what an agent is, we may nevertheless state that many of the Internet/Web applications involving agents, make use of the agent-metaphor only. In our opinion, such applications could benefit from employing actual agent technology, instead of merely using the metaphor. In Section 6 we will briefly outline what needs to be done to make such technology availabe to internet-based services. Here we will merely indicate the possible scope of applications which can deploy agent technology on the Web.

In [14] an overview is given of agent research and applications developed with agent technology. When we restrict ourselves to Web-related agent applications, we encounter information management, e-commerce and entertainment as the most dominant application areas. Most of the adaptive systems described in [5] only use the agent metaphor but are not really agent-based.

Since our WASP project concentrates on information management we first distinguish between a variety of tasks in this application domain where agents could be deployed:

- information retrieval – to collect and filter information
- information presentation – te present the information in an understandable way
- intelligent navigation – to enable the selection of relevant information

In order to realize such functionality in a Web-based context, a framework of agent technology is needed. Apart from the programming aspects, such a framework must allow for a declarative description of agent-behavior based on user preferences, including the cooperative behavior of software agents, that is how they interact with other agents.

In Section 6 we will indicate the research issues involved in developing such a framework. In particular, we will concentrate on the realization of such agents in the context of virtual environments, which (as we will argue in Section 5) imposes additional constraints and leads to additional requirements such a framework must satisfy.

3 Agents in virtual environments

Virtual environments, in particular 3D virtual worlds, are becoming a realistic architecture for providing services on the Web. 3D worlds offer more possibilities for presenting large information spaces in an attractive way than 2D presentations. As a consequence, agent-mediated services need an adequate representation in such environments, In this section, we will establish what possible function agents have in virtual environments, by briefly looking at the history of agent manifestations in such environment, and by exploring how agent technology can be deployed in virtual multi-user communities for helping visitors in information retrieval tasks and navigation. In summary, we may encounter agent-like creatures in the following contexts:

- autonomous characters in games

- avatar guides in virtual worlds

In Section 4, we will discuss autonomous characters in games. In this section we will explore how agent technology may play a role in general-purpose virtual environments.

3.1 Agents in the blaxxun Community Server

In many (3D) virtual environments, when entering the world, or a particular space, the visitor is welcomed by some 'officer' or guide. In most cases the welcome is reflected in the chat panel, announcing the new visitor to the other visitors. Examples of such guides can be found in AlphaWorld [1] and environments created with blaxxun Community Server technology [4]. The blaxxun Community Server is a state-of-the-art platform for creating virtual environments. Apart from the server, which maintains information about the state of the community and the whereabouts of its visitors, virtual worlds are defined using VRML-defined 3D worlds.

When entering the community, members or visitors may choose for 3D chat mode, which means that they find themselves in a virtual environment in 3D space, surrounded by objects, and possibly by other visitors and the agent or guide that belongs to that particular space.

The blaxxun Community Server does provide support for agents. Agents in the blaxxun Community Server may be programmed to have particular attributes and to react to events in a particular way. As a remark, originally the blaxxun agents were called bots. In our opinion the functionality of blaxxun agents does not surpass that of simple bots, and we consider the term agent to be a misnomer. Nevertheless, it is interesting to look in more detail how the behavior of blaxxun agents can be defined, and to explore how we may enhance the behavior of these agents in such a way that we can speak of 'intelligent agents'.

The blaxxun agent can be programmed using an event-based scripting language. An agent or bot script consists of four section:

1. robot definition – to define the robots nickname and choose a graphical representation
2. reaction definition – to define patterns of events and actions
3. session definition – to indicate to which scenes and which server the bot belongs
4. start session statements – for initializing specific sessions

The robot definition section allows for a rich repertoire of attributes, including roles, experience, interests and even a business card.

Reactions are defined by indicating an event or event pattern and a corresponding action. The blaxxun platform offers a rich set of built-in events, which includes both events coming from the chat channel as well as 3D events which occur in the associated 3D world. Actions include simple utterings, which appear in the chat channel as text, as well as movements of the avatar, which may beam to any desired spot in the world. Also timed actions are possible, which occur in response to a timer event, so that the bot or guide can periodically undertake some action.

Despite the large number of built-in events and the rich repertoire of built-in actions, the blaxxun agent platform in itself is rather limited in functionality, simply because event-action patterns are not powerful enough to program complex behavior that requires maintaining information over a period of time.

Luckily, there is a way out. That is, the blaxxun agent platform also offers a C++ API which allows the developer to augment the functionality of the simple bot. This puts the burden on the shoulder of the developers, who are faced with the task of creating their own platform for intelligent agents in virtual environments. Still, it offers a presentation and interaction basis that may make the development of a complete "agent support system" somewhat easier.

3.2 Information retrieval in virtual environments

When the question arises what possible use agents may have in virtual environments, we may think of information retrieval along the lines as indicated in section 2 as one possible and interesting example.

Basically, the information retrieval problem in virtual environments consists of detecting containment relations between worlds, and gathering information about the objects in that world, which includes geometric shapes as well as text, image and sound objects.

We may approach this problem in three radically different ways:

- using black-box feature detectors
- meta information
- incremental knowledge gathering

Black-box feature detectors

Feature detectors are a common means in multimedia database systems to obtain information about particular media items, such as images and audio in order to establish a measure of similarity between such media objects.

Despite problems with respect to selecting a relevant set of features and assigning proper weights in determining the similarity of media objects, feature detectors are effective when dealing with binary encoded material such as images and audio. When considering the material virtual environments are made of, however, we must note that much of the contents of virtual worlds is programmed content, that is the result of algorithmically generated geometries. Apart from that, it is hard to see how to obtain information about a world from a purely geometric description. In our opinion, feature detectors may be useful to inspect parts of the content of virtual worlds, in particular image or movie textures and sound objects. This information may then be used to augment the information obtained about a virtual world using any of the alternative approaches.

Meta information

Meta information is an effective way to allow searching for media content. Instead of describing content by intrinsic features of the media object, meta information is used to indicate properties of media objects that can often not be extracted from the media object itself. See [25]. As an example consider an audio recording of an aria. Information concerning the composer and singer is most conveniently stored as meta data, although it would in principle not be impossible to extract such information algorithmically.

Meta information in virtual environments would mean additional descriptive information that describe the content and/or context of a particular world. In VRML 2.0, we can use the WorldInfo node to provide such information. Also it is conceivable to generate such information dynamically, based on user ratings.

Incremental knowledge gathering

As noted before, a problem with virtual environments is that they are often generated in an algorithmic fashion, using information stored in a database. Moreover, the geometric information contained in a VRML file is not what makes the world interesting to a user. Rather, the interactive facilities provided by such worlds, and the navigation behavior of (other) users are what we are interested in. So, instead of a static analysis of virtual worlds, we believe that a more incremental approach is desirable, an approach where the navigational behavior of the individual user contributes to what we may call the collective awareness of a world. In other words, when we record the navigational behavior of a user and store the features of the world that are relevant to this user in a database, other users may profit from this information and query the database to be guided to the locations of interest.

3.3 Adaptive environments – presenting information

Given support for information retrieval in virtual environments, the question arises how to present this information to the user. One possibility, although not very appealing, would be a list of alternatives. For example, when the user asks for locations where there is information about, let's say patterns in object-oriented programming, a list of links to such information sources may be presented, as is common for most search engines. Then selecting a link might result in being ported to a world containing such information or any other location on the Web.

Alternatively, and more appealing, one may generate a virtual world that contains the answers to the query located in 3D. A clear advantage of generating a world in response to a query is that the topological properties of the world can be employed to sort the information according to some criterion of relevance.

Clearly, we have to distinguish between queries that concern information that is present within a world, queries that concern related worlds (that is worlds that are accessible from the world from which the query is posed), and queries that concern material that lies outside of the world (such as the request to locate paintings of the 17th century painted in a particular style). In the first two cases a direct transfer to a location of interest might take place, provided the location is available. In the latter case, we can only resort to generating a world based on the information that is found. Note that taking this approach to the extreme would mean to generate worlds based on the interest of visitors from the start.

Whatever approach is chosen for presenting the results from a query, the amount of information presented to the user is usually overwhelming. So, in addition to a world which contains that information, it is desirable to have a guide or agent that shows the user around, that is gives information about the results, indicates its relevance and transports the user (in a quite literal sense) to the area of interest. Such an agent could be augmented with information concerning the user's interests, (knowledge) background and preferences.

3.4 Discussion

Despite the availability of virtual worlds, we must remark that the notion of virtual worlds itself is somewhat elusive. Restricting ourselves to (3D) multi-user virtual communities based on blaxxun Community Server technology we may distinguish between (e)commercial environments, educational environments or environments offering a mixture of commerce and entertainment.

With regard to these environments we observe that

- many environments are rather static,
- most worlds are far less immersive than desirable,
- interaction facilities are generally poor, and
- navigation is often a problematic issue.

Coming back to the question what use agents might have in virtual environments, we may take this list to develop a program of research to tackle the issues involved.

Dynamically generated worlds

Virtual worlds may be generated based on a user's profile, to contain the information that corresponds with that profile in accord with the style preferences indicated by the user.

Immersiveness

The lack of immersiveness is to a large part due to for example chat windows and property menus, in other words 2D gadgets that are needed for customization and interaction. Although agent technology has no contribution to make in this respect, it is conceivable that the user's avatar is made more intelligent and is augmented with features that makes the additional gadgets obsolete, for example text balloons for displaying chat and instruments for customization.

Interaction

Experience shows that navigation and interaction are difficult (for the average user), simply because 3D interfaces are more complex than there 2D counterparts. To aid the user in learning how to use a world the effectively, we can think of helper agents that shows the user how interaction and the manipulation of objects takes place.

Navigation

Due to the inherent visual complexity of (3D) virtual environments, navigation remains a problematic issue. Fixed viewpoint animations do exist, but are not sufficient to direct the user to locations of interest. Instead, we should consider navigation by query, possibly augmented with intelligent guides that accompany the visitor on touring the world.

4 Autonomous characters in games

Although not widely accepted as science, the development of computer games has fueled the development of many state-of-the-art techniques.

Computer controlled opponents have always been present in computer games and used to consist of only very basic logic that enabled them to mimic more complex behavior. These simple computer controlled opponents where usually not very intelligent, most of these characters simply followed a predefined path through the play-field or had some sort of simple algorithm which allowed them to chase the player.

In recent computer games, however, the development of more realistic virtual environments in which the games take place has also created a demand for more intelligent behavior by the computer controlled inhabitants of the game world. These autonomous computer controlled characters can be seen as autonomous agents intended for use in game-based virtual realities.

4.1 Some applications of agent technology in games

There are several types of games that have a use for agent technology. In fact, looking towards gaming as a test-bed for agent technology seems to be an appealing idea. For example, the RoboCup [22] initiative is such an agent technology test-bed based upon a international soccer competition.

Games containing artificial life are becoming popular lately. One can think of the Tamagochi [2] hype of a few years ago and the recent Pokemon [19] hype. These games require players to raise and support any number of virtual pets. These virtual pets could be viewed as agents in the sense that their development depends on both their internal state and interactions with the environment, see [8].

A more complex kind of agent appears in the so-called 'interactive story' (or movie), which is a type of game where the user takes part in a story and the users actions influence the development of the plot. Since these games put a lot of emphasis on the drama, the agents should not only be able to navigate properly through the world, but they also need to act as realistically as possible. In [11] a system to implement a community of interacting agents for interactive stories is presented.

But perhaps the most interesting application of agent technology in computer games concerns virtual reality. As personal and dedicated game computers have become rather powerful, (realistic) 3D graphics in games is more rule than exception. Within these environments, realistic behavior of the computer controlled characters has become an important issue. On the one hand we have flight simulators and other more serious applications of this technology, on the other hand, more interesting from our point of view, we have the immensely popular 3D action-adventure games.

4.2 3D action-adventure games and virtual communities

3D action-adventure games (pioneered by idsoftware [13] in their by now classic game Wolfenstein-3D) has players walking around in a virtual world, shooting at opponents and, in more adventurous variations, solving puzzles. A recent game of this type, Quake 3 [20], may be regarded as the prime example of this type of games.

3D adventure games, especially when played online, may be regarded as restricted versions of online communities, with entertainment as their sole intended purpose. Players are embodied as humanoid avatars or, more commonly, as monstrous alien creatures.

Players can walk around in the virtual world, see the virtual representations of other players, chat with each other through text and even make gestures to each other. Interaction with the environment itself includes the ability to manipulate items and push buttons or fire a gun. The entertainment aspect of these worlds mostly comes down to the killing of ones opponents. However, avoiding moral judgement, social interaction is evidently an important aspect of these worlds. Social structure and rules of behavior are simply different. And one must recognize that the repertoire of interaction in these games, although apparently limited, is actually quite advanced when compared to the more generic online communities on the web.

Types of agents An excellent example of a virtual guide can be found in the game halflife [12, 23] where a computer controlled character guides the player through the training courses and teaches the player how to overcome various obstacles that can be expected in the actual game. But due to the repetitive and pre-programmed nature of these guides they are probably not worth te be considered as agents.

A more interesting kind of computer controlled character that occurs in action-adventure games consists of the computer controlled opponents. These can vary from monsters encountered in an adventure scenario, where their 'intelligence' is limited to attacking when the player approaches their residence, to the so-called 'bots', who actually pose as human opponents and have a wider freedom in the actions they may take. Note that computer controlled characters of this kind face many of the same problems as agents in other online communities, especially in the field of navigation.

Determing a course of action Although computer controlled characters can be quite complex the reasoning underlying their behavior can be split up into a few distinct steps. Roughly, the reasoning involves subgoal determination, planning, and navigation:

- (Sub)Goal determination: which can be anything from retrieving a certain object to attacking of even fleeing from an opponent.
- Planning: plan a route through the world to reach that particular objective.
- Navigation: navigate the planned route, taking into account obstacle avoidance and evasion of possible threats.

Reasoning is an iterative process, which means that at any time an agent might need to alter its current objective in favor of a more urgent one and change its course of actions accordingly. For instance when an agent is attacking an opponent and finds itself outgunned, it might decide to flee in order to survive.

Goal determination is a classic artificial intelligence problem and many techniques have been developed. A hierarchical approach is in these cases the most suitable[3]. The top level goal is to win the game. In order to achieve this goal there are two important subgoals. Firstly, kill your opponents and secondly try to survive. Since no exact prediction can be made on the state of the world in the future there is no way to determine the perfect course of action and short term sub-goals are constantly devised and reevaluated. Due to the obvious need for efficiency most of this reasoning is usually implemented through fairly simple decision rules.

Route planning is a complex issue. The fact that these virtual worlds are 3D and that there are numerous constraints that are to be met makes the route planning algorithms as used in these kind of games rather complex. For instance if an agent is wounded, it might not want to pass through parts of the world where it can expect to be attacked. Or, if an agent is chasing an opponent, it might want to out-whit its opponent and take another route to the expected destination of the fleeing opponent in order to set up an ambush.

Finally, navigation itself poses some problems. Knowing the route to be followed in order to reach a certain location is not enough to travel through the world safely [21]. An agent needs to take into account the (virtual) physical limitations of its embodiment, for instance not even a computer controlled character is allowed to pass through walls. In addition to this, the possible need to make evasive manouvres to avoid being shot possibly in combination with the need to actually aim at an opponent causes even more navigational difficulties. Also note that not all obstacles that may need to be avoided are known in advance, since one can simply not predict the actions of other players.

5 A taxonomy of agents

Different aspects of applications and configurations suggest different types of web agents. In this section, we will propose a taxonomy of web agents. With respect to the choice of dimensions, we must remark that even though our taxonomy has a bias towards the deployment of agents in virtual environments, the taxonomy is valid with respect to Web-based services in general.

We consider the following three dimensions of web agent types.

1. 2D versus 3D
   A 2D web agent is one which is aware of http, file, and ftp protocols, whereas a 3D web agent is one which is aware of not only those protocols, but also virtual reality protocols. Typical 2D web agents provide a text-based interface, for example, information retrieval services. Typical 3D web agents are avatar-embodied guides that help visitors to navigate in virtual environments.

2. Client versus server
   As the names imply, a client web agent is on client side, whereas server web agent is on server side. A typical client web agent can serve as a personal information assistant. A typical server web agent can serve as the front-end of web servers to offer information more intelligently.

3. Singularity versus multiplicity
   As the names imply, a single web agent would not consider the interface with other web agents, whereas multiple web agents would interact on each other.

There are different types of web agents based on those three dimensions, which consists of a complexity lattice of web agent types, which is shown in Figure 1. 3D-server-multiple-agent is on the top of the complexity hierarchy, whereas a 2D-client-single-agent is on the bottom. Those dimensions are not exhaustive. In particular, in this paper, we do not consider the dimension "stationariness versus mobility"[1]. However, the combinations based on the dimension "stationariness versus mobility" can be generalized accordingly, based on the current taxonomy. We leave it as further work. All the dimensions we consider for the

[1] A stationary agent is one that executes only upon the agent platform where it begins executing and is reliant upon it, whereas a mobile agent ts one that is not reliant upon the agent platform where it began executing and can subsequently transport itself between agent platforms[10].

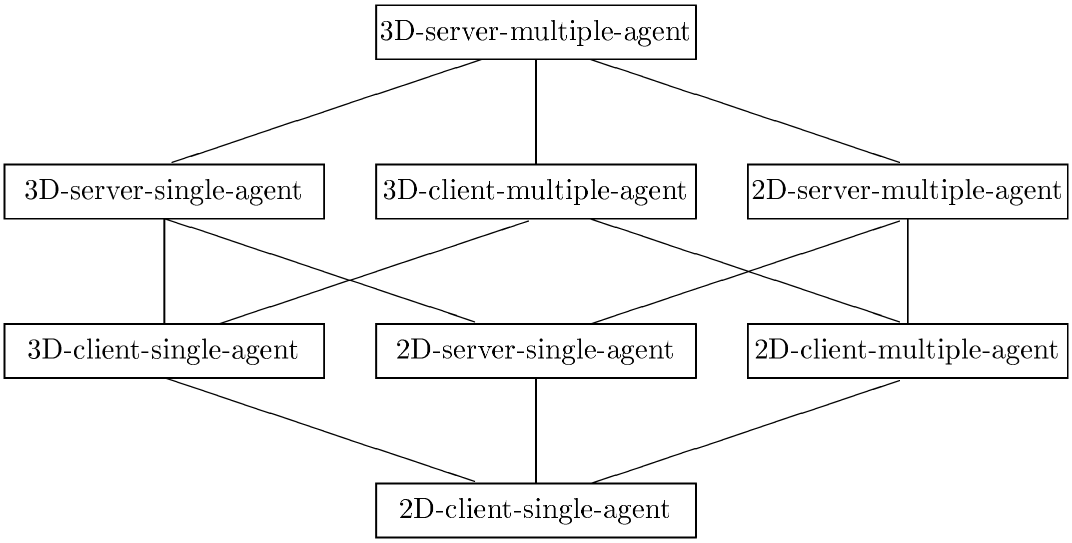

Figure 1: Lattice of Web Agents

taxonomy are directly web-dependent. The dimension "2D-3D" specifies the internet protocols agents have to be aware of, the dimension "client-server" is concerned with internet service modes agents have to offer, and the dimension "singularity-multiplicity" decides agent communication languages. We do not consider the functional dimension which are not directly relevant to the Web, like cooperating agents, problem solving agents, and negotiation agents, etc. We do not discuss the dimensions which are domain dependent, like expertise seeking agents, e-commerce agents, etc. However, our taxonomy does cover most types of those agents. They are either 2D agent or 3D agent, either client agent or server agent, either single agent or multiplity agent, or some of their combinations. Different types of web agents suggest different types of interaction modes with users and web servers. We discuss them in detail in the following.

3D-Server-Multiple-Agent

The configuration of 3d-server-multiple-agent on the Web is shown in Figure 2. The agents are the parts of virtual reality servers, more exactly, virtual community servers, for they interact with multiple users and agents. Users communicate with virtual reality servers via web browsers. The agents interact with each other in the virtual reality server, and communicate with web users. As agents in virtual reality, they are normally embodied by their avatars.

As discussed in Section 3 and Section 4, these kinds of web agents have a great deal of application potentials, which range from avatar-embodied guides that help visitors in information retrieval tasks and navigation in virtual environments, e-commerce shopping assistant agents in 3D-virtual shops, to intelligent agents in multiple player computer games.

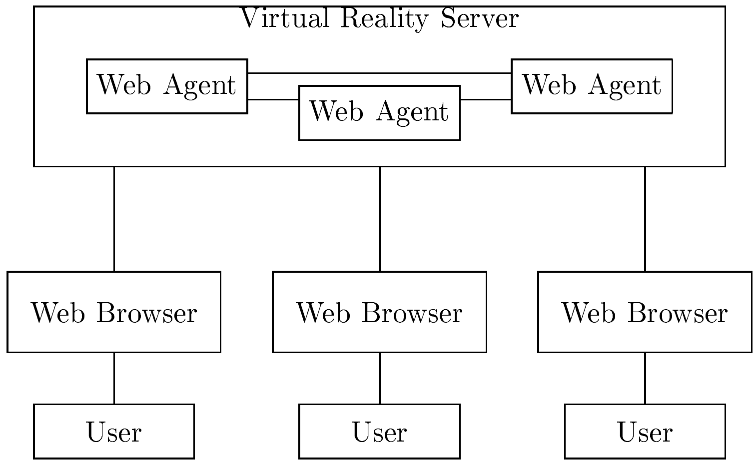

Figure 2: configuration of 3d-server-multiple-agent

3D-Server-Single-Agent

3D-server-single-agents and their relationship with servers and users are shown in Figure 3, where agents serve as the front-end of the servers. Users do not directly communicate with servers, but with the intelligent web agent which is located on the server side. In other words, the agent may be regarded as a portal through which the server os accessed.

That kind of web agents can intelligently offer or create personalized virtual environments for users, based on users' queries or requests.

3D-Client-Multiple-Agent

The relation among 3d-client-multiple-agent, virtual reality server, and web users is shown in Figure 4. The agents are located at the client side, normally serve as users' personal information assistants. Virtual reality servers may have their own web agents. Web users communicate with servers or other users (or agents) either via web browsers or directly via web agents. Other users may also have their own web agents (which are omitted in the figure for the reason of the simplity). This type of web agents is particularly useful in the 3D chat arena, in which the agents can serve as intelligent personal assistant to help users in information gathering, and interface with other intelligent web agents.

2D-Server-Multiple-Agent

The configuration of 2D-server-multiple-agents is shown in Figure 5. Similar as its 3D counterparts, these kinds of web agents are a component of web servers. They can offer users a text-based or image-based interface for the navigation. One of the useful application for them is to serve as intelligent e-commerce sales-bots to offer more user-friendly and more intelligent services. This type of web agents can display intelligent behaviour in text-based internet games, like MUD games[16].

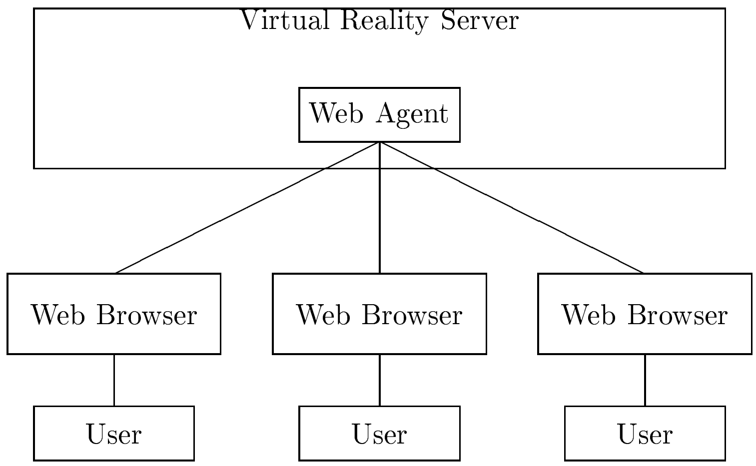

Figure 3: configuration of 3d-server-single-agent

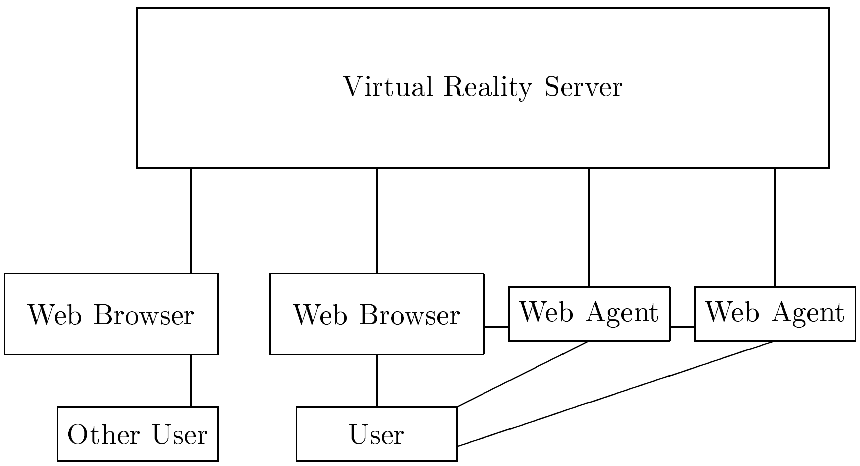

Figure 4: configuration of 3d-client-multiple-agent

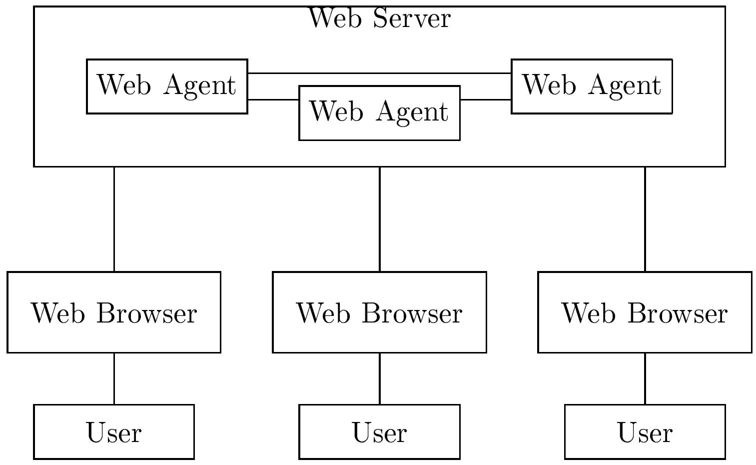

Figure 5: configuration of 2d-server-multiple-agent

3D-Client-Single-Agent

3D-client-single-agent is a simplified 3d-client-multiple-agent. Its relations with other web components is like the situation shown in Figure 4, except that there is only one agent available. Similar to their multiple-agent counterparts, this type of web agents normally serves as personal information assistant in virtual environments.

2D-Server-Single-Agent

2D-server-single-agent, which is a restriction of the 3D-server-single agent, could alternatively be called *intelligent web server*. These kinds of agents normally locate at the front end of a http server to offer users more personalized messages. They can serve more efficiently and more intelligently if they are supported by powerful inference engines.

2D-Client-Multiple-Agent

The configuration of 2D-client-multiple-agent is analogous to the 3D-client-multiple-agent which is shown in Figure 4, apart from the more powerful user interface and display properties of the latter. One useful application of them is to act as an internet chat agent. They help users to record and analysis chat messages, to gather information and interact with other users or agents when requested.

2D-client-Single-Agent

2D-client-single-agents may simply be called personal (text-based)-information assistants. Its relation with servers and users is analogous to the situation shown in Figure 4, except that there is only one agent, which only has a 2D manifestation. Although they are among the simplest form of web agents, as they lack the facilities to interact with other web agents, they can still fulfill a lot of interesting and useful tasks, like gathering information on request.

6 Research issues

With the taxonomy as outlined in the previous section, we are able to indicate the research issues involved in developing a Personal Assistent for Maintaining Electronic Archives (PAMELA). We note that the PAMELA application is just one in a range of possible agent applications for the Web, and we will mention some more applications at the end of this section.

The primary requirements PAMELA must fulfill may be summarized as follows:

- autonomous and on-demand search capabilities
- (user- and system) modifiable user preferences, and
- (multimedia) presentation facilities

Evidently, with regard to our taxonomy, we have a choice of where to put the functionality, i.e. either on the client-side or server-side, whether we support a 2D (text) interface or an embodiment in 3D, and whether we realize PAMELA as a single agent or as a multi-agent system. Each of these choices has ramifications with respect to the effort of developing the application as well as the resources needed for deploying it.

Nevertheless, irrespective of a particular choice, the following aspects play a role in developing Web-aware agents, or for that matter, a framework for developing such agents:

- generic agent software components
- user interface components for managing agents
- communication primitives for agent communication

Where we position the application in our lattice of possible agent applications will, however, affect the complexity of the components mentioned above. For example, developing 3D-client-multiple agents will require advanced interface components, as well as powerful communication primitives.

To deal with the complexities involved in developing a range of agent applications, we aim at providing support on the level of the software architecture. Architectural support for agent applications must, in our view, include

- an agent-based programming language
- a high-level API for Web-aware applications
- an object-based framework for distributed agent applications
- tools for agent-based information retrieval and management

Actually, in order to offer more than the agent metaphor as a guideline for developing applications, we think it is necessary to deploy an agent-oriented programming language supporting the primitives needed for expressing both the reasoning and the domain or context in which this reasoning takes place. The ingredients for such a language, which must be distributed, object-oriented and logical are outlined in [9]. In addition to the language, an agent-framework must provide components for information retrieval, communication and presentation.

	2D-agent	3D-agent
blaxxun agent	chat	avatar
blaxxun server	–	world
blaxxun client	–	EAI

Figure 6: Agents in the blaxxun platform

Agents in virtual environments Having an agent-oriented programming language as the core of our framework, however, does not solve all problems, especially not when we aim at developing 3D-server agent applications, which must be integrated with for example the blaxxun Community Server. Apart from components for information retrieval, information storage, Web access and communication, we must also specify how the particular communication structure of the community server, which uses client- and server-generated events as the primary communication mechanisms can be adapted to allow for delegating the agent-related issues to components build with our framework. Technically, this requires the development of:

- the definition of an extension API for the blaxxun Agent, and
- the development of an additional application server, that intercepts client-generated events.

More specifically, taking the blaxxun Contact client (which is a generic VRML browser with support for multi-user shared events) and the blaxxun Community Server platform as our point of departure, we may distinguish according to our lattice between agents that are either 2D or 3D and, independently, either located on the server-side or client-side. In addition, we have the single/multi-agent dimension, which we ignore for the moment. Apart from the community server, the blaxxun server platform also offers the blaxxun agent or bot, that may be used to provide intelligent services as well.

Figure 6 summarizes how an agent framework may interact with the blaxxun (client and server) platform. As concerns the blaxxun agent, we may have a 2D (single or multi) agent that acts on the server-side using the chat interface, or a 3D agent that may be represented as an avatar in the 3D world. Both types of agents require an extension of the blaxxun agent API with (preferably) intelligent facilities that aid the user in search and navigation. As an example, the user may employ the chat interface to ask the agent for information or, alternatively, direct the agent's avatar in 3D space to undertake some action, for instance by dropping an icon on the avatar.

In general, agents that live on the server-side may directly influence or modify the 3D world. For 2D agents, it is hard to find any suitable form of manifestation in such a world, apart from text or sound. As an example of a 3D-server-side agent, think of a virtual broker that may show you around in (the model of) a house that you might want to buy. Support for such agents must be provided by extending the event-dispatching mechanism on the server-side with an additional application server, that may intercept (client) events and generate additional (server) events.

Finally, we have to consider how we may deploy agents in the (blaxxun) client, which we use for visiting virtual environments. Recall that VRML offers an External Authoring Interface (EAI), which allows for programming a VRML world using Java. The EAI may

	type	blaxxun platform
lexicon search	2d-*-server	agent
information gathering	2d-*-server	agent/server
information presentation	3d-*-*	client/server
information visualisation	3d-*-client	client/server
navigation	3d-single-server	agent/server
brokering	*-*-*	agent

Figure 7: Demonstrators of agent technology

be used to get access to agent functionality, including multi-agent cooperation, provided that such functionality is available in Java. As an example of how client-side agents might be useful, we may think of delegating the actual presentation of information gathered by a server-side agent to a (client) that resides in the (VRML) browser.

Demonstrators for agents in virtual environments Developing agent-technology for virtual environments is a challenging task, but probably an endeavor that is doomed to fail when one does not focus on target applications or demonstrators that deliver some useful service that can be tested in practice. To conclude our discussion of research issues, we will briefly present a list of possible target applications that may serve as the agenda for research to come.

Figure 7 presents a number of possible applications of agent-technology in virtual environments, with an indication of the type of agent involved, as well as an indication what part of the blaxxun platform is affected. An asterisks (*) is used as a wildcard to indicate where any of the alternatives is possible. As an example, for information visualisation, we will very likely need an agent that operates at the client-side, even though it uses other agents that operate at the server-side. As another example, to define a brokering service that allows visitors to come in contact with another visitor or service, any choice in the agent lattice given in Section 5 is conceivable. However, when providing navigation services, the range of alternatives is limited to one, namely 3D-single-server agents, although also in this case a multi-agent solution would be conceivable as well.

To realize any of these target applications, significant endeavor is required to model the services that are provided and for embedding these services in the blaxxun platform in a more or less seamless way. As indicated in Section 3.1, we strive for the realization of immersive virtual environments, which means that the structure and presentation of a virtual world must not be disrupted because of the availability of some service, however useful it may be. To find a seamless integration between intelligent (agent-based) services and 3D virtual worlds may, in other words, be taken as the presupposition of our research, if not a research goal in itself.

7 Conclusions

In this paper we have presented a taxonomy of agents on the Web. However, to be honest, our taxonomy was geared towards the deployment of agents in virtual environments. Particularly, the agent-lattice resulting from our taxonomy implies on ordering on conceivable agents along the dimensions 2D or 3D presentation, single or multi-agent support, and client or server-side residence. This taxonomy has been applied to indicate research issues and the ramifications target applications have with respect to a platform for realizing intelligent services in a virtual environment, in particular the blaxxun Community Server platform and the blaxxun (VRML) client.

References

[1] Alpha Enterprises, World of Alpha, http://www.alphaworld.com, 1999.

[2] Bandai America Incorporated website, 1997. http://bandai.com.

[3] Bell, J., and Huang, Z., Dynamic goal hierarchies, in: L. Cavedon, A. Rao, W. Wobcke (eds.), *Intelligent Agent Systems, Theoretical and Practical Issues* , LNAI 1209, Springer, 1997, 88-103.

[4] Blaxxun Interactive website, http://www.blaxxun.com, 1999.

[5] Brusilovsky, P., Methods and Techniques of Adaptive Hypermedia, *User Modeling and User-Adapted Interaction*, 6, 87-129, 1996.

[6] Caglayan, A., and Harrison, C., *Agent Source Book – a complete guide to Desktop, Internet and Intranet Agents* , Wiley, 1997.

[7] Cheong, F-C.,*Internet Agents: Spiders, Wanderers, Brokers and Bots*, New Riders, 1996.

[8] Creature Labs website, 1999. http://www.creaturelabs.com.

[9] Eliëns, A., *Principles of Object-Oriented Software Development*, Addison-Wesley, 2000.

[10] FIPA, FIPA Content Language Library, Foundation for Intelligent Physical Agents, 1999.

[11] Graham, M., and Wavish, P., A situated approach to implementing characters in computer games. *Applied Artificial Intelligence*, 10(1):53-74, 1996.

[12] Official Half Life website. http://www.sierrastudios.com/games/half-life, 1999.

[13] ID software website, http://www.idsoftware.com, 1999.

[14] Jennings, N.R., Sycara, S., and Wooldridge, M., A Roadmap of Agent Research and Development, *Autonomous Agents and Multi-Agent Systems I*, Kluwer, 275-306, 1998.

[15] Maes, P., Humanizing the global computer, Interview in: *IEEE Internet Computing*1(4), 1997.

[16] Mud Connector, http://www.mudconnect.com, 2000.

[17] Negroponte, N., *Being Digital*, New Riders, 1995.

[18] Petrie, C., What's an agent ... and what's so intelligent about it?, WebWord column, *IEEE Internet Computing* 1(4), 1997.

[19] Pokemon World, the official pokemon website, 1999. http://www.pokemon.com.

[20] Quake III Arena website, http://www.quake3.com, 1999.

[21] Reynolds, C. W., Steering Behaviors for Autonomous Characters. March 19, 1999. http://www.red.com/cwr/steer/gdc99.

[22] RoboCup, The Robot World Cup Initiative, http://www.csl.sony.co.jp/person/kitano /RoboCup/RoboCup.html, 1999.

[23] Sirerra Studio website, http://www.sierrastudios.com, 1999.

[24] Subrahmanian, V.S., *Principles of Multimedia Database Systems*, Morgan Kaufmann, 1998.

[25] Watson, M., *AI Agents in Virtual Reality Worlds – programming intelligent VR in C++*, Wiley, 1996.