Multi-Agent Systems for Web-Based Map Information Retrieval[1]

Maozhen Li, Sheng Zhou and Christopher B. Jones

Deptartment of Computer Science
Cardiff University, Cardiff
CF24 3XF, United Kingdom
{maozhen, s.zhou, c.b.jones}@cs.cf.ac.uk

Abstract. Many types of information are geographically referenced and interactive maps provide a natural user interface to such data. However, map presentation in geographical information systems (GIS) and on the internet is closely related to traditional cartography and provides a very limited interactive experience. We present the MAPBOT, a web-based map information retrieval system to search geographical information using software agent techniques. Each kind of map feature such as a building or a road is treated as an agent called a Maplet. Each Maplet has a user interface level to assist the user to find information of interest and a graphic display level that controls the presence and the appearance of the feature on the map. Semantic relationships of Maplets are defined in an Ontology Repository (OR) and used to control their retrieval. To allow system developers to manipulate the OR, an Ontology Editor is implemented with a graphical user interface. Visualisation on the client is based on Scalable Vector Graphics (SVG). The paper describes the current state of development of the prototype and presents, and evaluates experimentally, an agent-oriented graphical conflict resolution procedure, which demonstrates the potential for conflict resolution via low level agent communication, as opposed to global control.

1. Introduction

Maps are being used increasingly in local, networked and mobile information systems for communicating geographically referenced information. This has become possible because of the now relatively widespread availability of digital map data and developments in geographical information system (GIS) technology. The applications are wide ranging including local government planning, environmental monitoring, market analysis, navigation and public access to information.

Interaction with a digital map is typically based on a cycle of elicitation of user input via menu and dialog boxes, selection of map areas or features, and return of

[1]This research has been funded by UK EPSRC grant GR/N23172

information, which may in turn induce modification to the map content. The maps themselves are often close replicas of traditional paper map cartography. The approach is to be found in many commercial GIS and is now being reflected in mapping applications on the internet. Developments in human computer interaction with regard to information retrieval and data visualization raise the question of whether the conventional approach can be improved. Certainly there is a motivation to investigate new approaches, since the current map interface, particularly on the Internet, often suffers from poor legibility of symbols and text, unnecessary user actions and inadequate adaptation to user interests.

Recent years have seen a marked interest in agent-oriented technology [1,2] spanning applications as diverse as information retrieval, user interface design, network management and complex system simulations [3,4]. An agent is a computational entity which acts on behalf of other entities in an autonomous fashion, performs its actions with some level of pro-activity and/or reactiveness, and exhibits some level of the key attributes of learning, cooperation and mobility [5]. The next generation of GIS may be expected to provide user interfaces with intelligent agents to assist users to search, query and operate the system [6]. The use of agents to represent map features is a characteristic of the project AGENT [7]. That project addressed the problem of map generalisation using software agents but was not directly concerned with information retrieval or interactivity issues. Here we present the MAPBOT, a multi-agent system for active maps which makes use of agent technology both for web-based map information retrieval and map generalisation. MAPBOT provides a user interface agent to assist users in effectively and efficiently searching map related information.

Finding the right piece of information at the right time can be quite a complicated task. Conventional information retrieval techniques can help us find relevant information, but interacting with these systems is often an awkward experience, especially for novices. User interface agents are adopted for finding contextually useful information appropriate to interactive tasks for users.

A user interface agent (UIA) is in charge of interacting with the user and of making the other agents transparent to him. The UIA is able to understand the user's requests and translate them for the other agents. It is also in charge of coordinating the work of the other agents [8, 9]. The interface agent performs a number of tasks that are crucial for the correct operation of an intelligent semantic matching system:

- assisting a user in performing requests and compiling his profile;
- deducing a user's information needs by both communicating with them and observing their "behaviours";
- translating the requests of a user and selecting the agent(s) able to solve their problem(s);
- presenting and storing the retrieved data.

A great deal of work has been carried out into the use of interface agents in recent years [10, 11, 12, 13, 14]. A number of prototype information retrieval systems have been developed that incorporate intelligent interface agents. Some notable examples are Ontoseek [15], InfoSleuth [16], RETSINA [17] and WebMate [18].

Many of the systems are based on ontology technology [19, 14, 15], they all rely on centralized ontology databases that are stored in relational database systems. The emergence of XML, OML and RDF allows the ontology metadata to be embedded in

the encoded web document, facilitating semantic matching by retrieval agents. Although the encoding agents may still refer to centralized ontology databases during the encoding process, the databases can now also be encoded in XML because of its openness. Like many systems, the UIA in MAPBOT makes use of ontology to validate user inputs and captures user interactive intentions, but the ontology in MAPBOT is organized in XML like OSCA [20]. In addition, the UIA works in a Web browser providing an easy use Web user interface for online map information retrieval.

A key concept of MAPBOT is an active map, characterised by the following features:

- Responds helpfully to direct manipulation of the displayed map symbols.
- Exhibits intelligence with regard to knowledge of concepts and context of individual map features and classes of feature.
- Guides the user to relevant related information within the map space and within the information space.
- Modifies its map symbols, icons and text dynamically to assist in focusing on particular types of information and particular geographical locations.

In MAPBOT, the active map will be associated with different agents: representing geographical phenomena, capable of communicating with the map user, visualized by symbolic or iconic representations that adapt to the scale and purpose of the map, and knowledgeable of their ontology, their spatial context and the application specific procedures that might be applied to them.

In the remainder of this paper we sketch out in Section 2 a scenario of the type of user interface facilities that we are aiming to achieve in the longer term. Section 3 describes the software architecture of the system, with reference the agent properties, functionalities and interactions of the Maplets that represent individual map features. This is followed in Section 4 by a summary of the current implementation which uses a Java servlet for web communications, the SVG standard for visualisation and a simple ontology in combination with a GUI-based ontology editor, to encode the user concepts associated with Maplets. Section 5 describes the results of some experiments which demonstrate the potential of using multi-agent techniques for resolving graphical conflicts in the map display. Here we focus on the specific case of simple polygonal objects and compare results with previous work on the same sub-problem. The approach is novel in the context of conflict resolution in being based essentially on message passing between neighbouring map features. Its effectiveness is demonstrated in comparison with two "global" iterative improvement techniques. The paper concludes in section 6 with a summary of results and a discussion of future work.

2. Scenario of Map-Based Information Retrieval

1. Suppose a user wants to travel to Cardiff. He downloads the MAPBOT from the internet, and a web page appears on his browser. There are a lot of iconic metaphors on the map with each one representing one category such as a bus station, a pub or a museum.

2. He types in Cardiff in the Search box, then a map covering the vicinity of Cardiff appears in the map window.
3. He moves the map in different directions to find something interesting.
4. He points to Transportation from the Common Category menu, then all information related to Transportation such as Railway Station, Bus Station and Cardiff Airport appears on the map. Other unrelated information like Rivers will disappear from the map, unless it is part of the transport infrastructure, or essential base map information. If he moves to a Bus Station and clicks the right button of his mouse on it, then an information menu will appear to ask him to select information of interest. Options might be *Timetable; How to get to the station; History of the station*. Then the system will lead the user to the related web page.
5. He types in a search sentence like " Tell me where is the nearest art museum to my current position?". After confirming the location, the system will pop up a window showing how the user could get to the museum and how far it is. The interface will understand some words like where, when, what, how far, nearest etc.
6. He moves his mouse cursor in the map window. The interface will display a message bar to show more detailed information about a map feature which is symbolized. For example, if he puts his mouse cursor onto a pub symbol, then a message bar will appear to show that this is a pub and where it is located. If he wants more information, he can click the right button of his mouse and then an information menu will appear to ask him to select more information.
7. When he moves his mouse cursor around the map, the map features that have a photo associated with them will flash continually. The user can point to one flashing object and click on it. Then a picture associated with the object will be displayed. When the user points to the picture and moves his mouse, the picture will move around and show itself from different angles to provide a Virtual Tour.
8. He types in "Interesting Places" in the Search box, then the map will display a range of types of places like a museum, a castle, a sports stadium and so on. Clicking on one of these results in the map displaying more of a similar type.
9. He can change his search by selecting on option from Common Category menu or typing in another search topic in the Search box.
10. The user interface agent builds up a profile of the user based on his interactive actions and, in the current and subsequent sessions, helps him to search for the information he needs, by anticipating his interests.

3. Software Architecture the MAPBOT

3.1 Agent attributes

According to Maes [22], an agent is a system that tries to fulfil a set of goals in a complex, dynamic environment. It can sense the environment through its sensors and act upon it using its actuators. The agent paradigm has been the subject of Artificial Intelligence (AI) research for some time, but it is only now that it is beginning to spread into information technology and commercial markets.

There are two main functions of agents in the MAPBOT. One is to provide an intelligent user interface for retrieving map related information, the other is to resolve graphic conflicts through the communication between map feature agents. Desirable attributes of agents used in MAPBOT are given below.

- Autonomous: Exercises control over its own actions.
- Reactive: Responds in a timely fashion to changes in the environment.
- Proactive: Acts based on goals, purposes.
- Interactive: Communicates with other agents, perhaps including people.
- Deductive: Deduces the user's interactive intention from their actions.

3.2 Agent types

There are many agents in MAPBOT. Based on their functionalities, the agents can be classified into User Interface Agent (UIA), Server Agent Broker (SAB) and Maplets. Each kind of map feature such as buildings and roads can be a Maplet. Figure 1 shows the software architecture of the MAPBOT. And we now explain the individual components.

User Interface Agent (UIA) receives inputs from a user and assists the user to search, query and manipulate the map in an efficient and user-friendly way. The UIA tries to understand the subject and geographical context of interest to the user and sends a message to the server to ask for more information or modify the map to change the content and level of detail. The UIA should keep a concise profile for each user to record their search interests. Server Agent Broker (SAB) receives requests from the UIA. It broadcasts a user's requests to Maplets. Based on the Ontology Repository (OR), the SAB can ask certain kinds of Maplet to display themselves or to provide more information to the user. Each kind of map feature is an agent represented by a Maplet. Each Maplet has its own control flow and can respond to change in its environment. Each Maplet has sensors to perceive its environment and actuators or effectors to act on it. The environment can be the system initialisation, SAB communication or other Maplet communication. Based on its reasoning rules, the effector can be "Display itself", "Disappear", "MoreAboutMe" or "MoveLocation". Figure 2 shows the control flow in a Maplet.

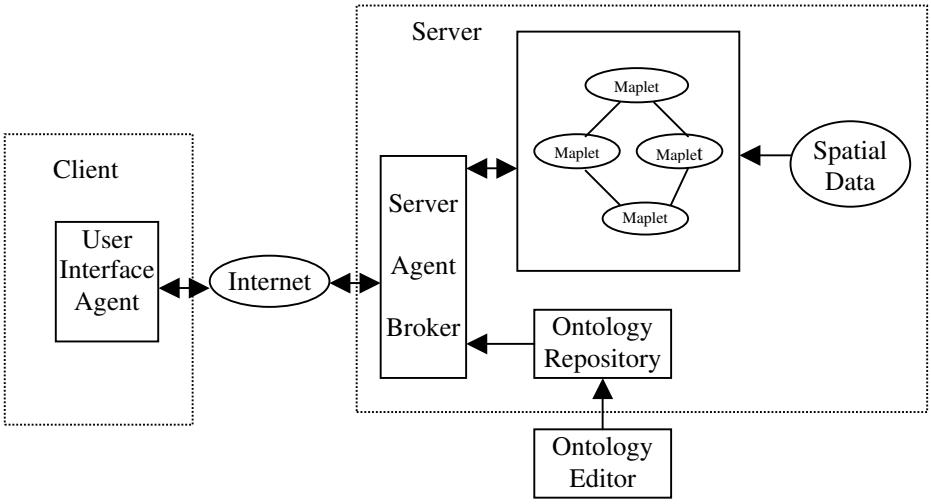

Fig. 1. Software architecture of MAPBOT

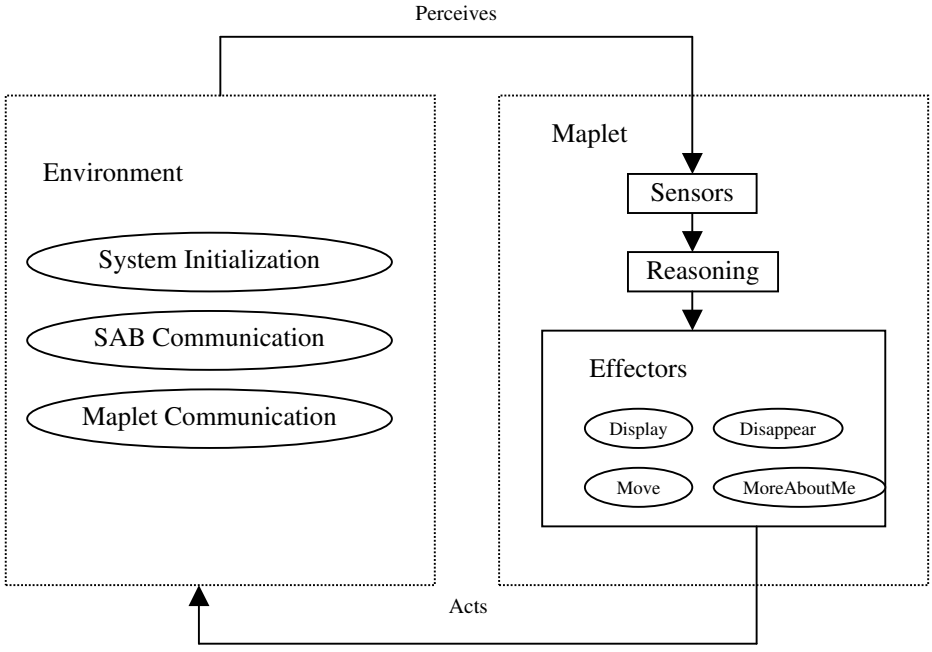

Fig. 2. The control flow in a Maplet

3.3 Agent interactions

In the following, we demonstrate a scenario of interaction among the MAPBOT agents in the context of a simple query execution.

During the system start-up, the SAB initializes itself and commences listening for queries from the UIA and for advertisement information from Maplets. Each Maplet registers itself with the SAB and retrieves related data from the Spatial Database. When registering with the SAB, each Maplet needs to tell the SAB the ontology concept to which it belongs.

A user starts interactions with the MAPBOT by means of a Web Browser, currently through Microsoft Internet Explorer. The user poses a query by means of pointing to a map feature of interest, typing the query in a search field, or selecting a query category from a category list presented in the applet. Then the UIA submits the query along with the profile to the SAB to retrieve information. When the user moves the mouse on a map feature on the map, the map feature is highlighted.

On receiving a request, the SAB searches the Ontology Repository (OR) to find matches with the query term, parses the profile to check if the query is valid and informs the UIA of the result. If the query is not valid, the UIA will display a dialogue to tell the user the query is not valid and ask them to perform another query. Otherwise the UIA will listen for further responses from the SAB to receive updated map data. The SAB retrieves the ontology information based on a valid query and broadcasts a request with the relevant ontology objects (concept terms) to all the Maplets registered with it.

After receiving the request from the SAB, each Maplet needs to decide to display itself or not. The ontology will assist in these decisions as it can be used to calculate a semantic distance between the query terms and the class of the Maplet. These distances will be used to rank the relevance of particular classes of information.

The SAB receives data from the Maplets that will be displaying (on the basis of exceeding a relevance threshold) and sends a formatted data stream to the UIA to update the map.

4 Implementation of the Prototype

4.1 The user interface

A MAPBOT prototype has been implemented in Java. The UIA is implemented as a Java Applet and runs on the client side to interact with a Web user. The SAB is implemented as a Java servlet using Tomcat 3.2.4 as the servlet engine. Each Maplet is implemented as a Java object. The visualisation on the client side is based on Scalable Vector Graphics (SVG) developed by World Wide Web Consortium (W3C) [21]. SVG has the following advantages:

- Plain text format - SVG files can be read and modified by a range of tools, and are usually smaller and more compressible than comparable JPEG or GIF images.

- Scalable - unlike bitmapped GIF and JPEG formats, SVG is a vector format, which means SVG images can be printed with high quality at any resolution.
- Zoomable - any portion of an SVG image can be zoomed in without any degradation.
- Interactive - SVG images can be made interactive through the assignment of event handlers, such as "onclick" and "onmouseover", to image elements. These can be used to highlight a selected map feature, show attribute information or display a map tip.
- Open Standard - SVG is an open recommendation developed by a cross-industry consortium. Unlike some other graphics formats, SVG is not proprietary.
- True XML - as an XML grammar, SVG offers all the benefits of XML.

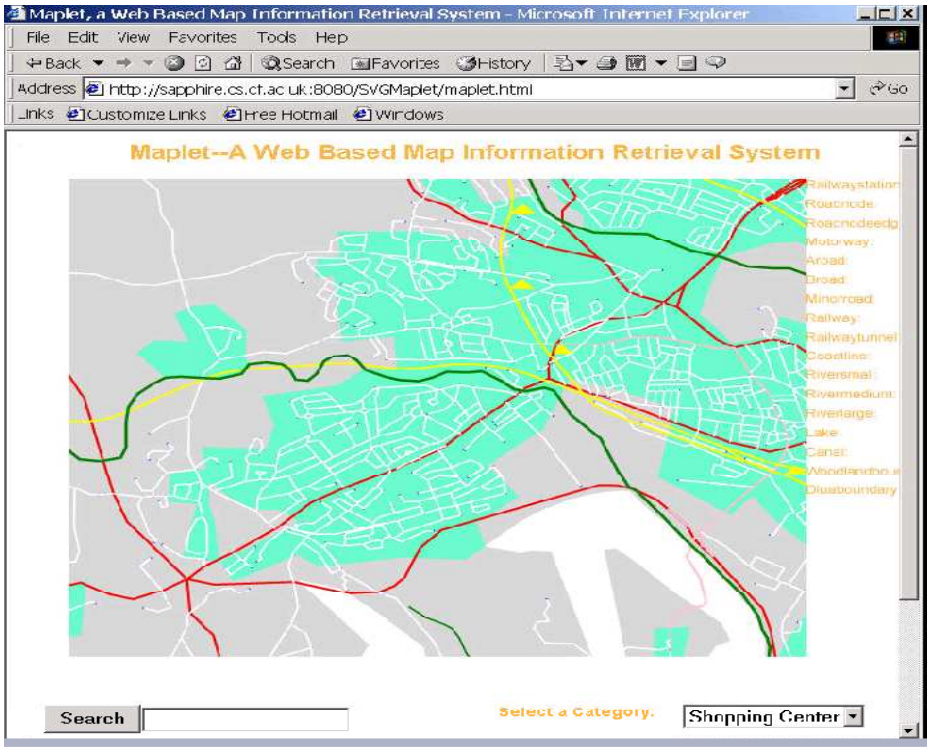

Fig. 3. A snapshot of the MAPBOT prototype (from source dataset OS Meridian-2 tile st28, © Crown Copyright Ordnance Survey. An EDINA Digimap/JISC supplied service)

To view the MAPBOT prototype, web users need to download a SVG Viewer plugin from Adobe [23] and use Internet Explorer. Figure 3 shows a snapshot of the prototype.

There are several components in the prototype, a web page, an invisible applet, a servlet and an ontology parser. The data workflow is given in Figure 4.

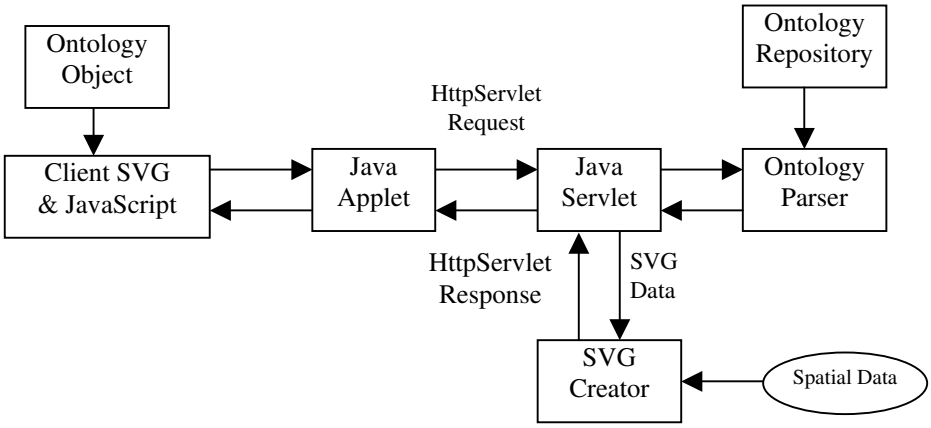

Fig. 4. The data flow in the prototype

When a user accesses the web page, the JavaScript embedded in the web page will call a "faceless" Java applet to send an http request to the Java Servlet which will invoke the Ontology Parser to create an Ontology Object from the Ontology Repository and then send the object as a serialized Java object to the applet. At present the Ontology Object contains the entire ontology. The applet uses the Ontology Object to check if the user has made a valid search. If valid, the Applet will submit the search to the Servlet which in turn invokes the SVG Creator to create a SVG file for the client to refresh the web page. A fragment of an SVG file created in this way is illustrated in Figure 5.

4.2 Ontology Repository

An ontology for a body of knowledge, concerning a particular task or domain, describes a taxonomy of concepts for that task or domain that define the semantic interpretation of the knowledge [24]. Ontology in the MAPBOT defines the semantic relationships of Maplets. The Ontology Repository (OR) is organised in a tree structure and described in XML. The DTD definition is shown below. The OR has one or more nodes in it. Each node has a name, a parent node, one or more synonyms and related keywords.

```
<?xml encoding="UTF-8"?>
<!ELEMENT map (node)+>
<!ELEMENT node (name, parent, synonym*, related*)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT parent (#PCDATA)>
<!ELEMENT synonym (#PCDATA)>
<!ELEMENT related (#PCDATA)>
```

```
  <?xml version="1.0" encoding="iso-8859-1"?>
  <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303
Stylable//EN"      http://www.w3.org/TR/2000/03/WD-SVG-
20000303/DTD/svg-20000303-stylable.dtd
   //standard SVG file header
  [ <!ENTITY railwaystationstyle "fill:yellow;">
    <!ENTITY roadnodestyle "fill:blue;">
    <!ENTITY lakestyle "fill:blue;">
    <!ENTITY woodlandboundarystyle "fill:white;">
    <!ENTITY                          dluaboundarystyle
"fill:rgb(100,255,200);">
    <!ENTITY dluaboundaryholestyle "fill:pink;">
  ]>
  //specify how to display Railway Station, Road Node,
Lake, Woodlandboundary and Dluaboundary.
  <svg   width="5.14cm" height="4cm" viewBox="310000 -
180000 10000 10000" enableZoomAndPanControls="flase">
  <rect     x="310000"     y="-180000"     width="10000"
height="10000" fill="lightgrey" stroke="blue"/>
  //Based on the map size, specify the viewBox size.
Adjust the Y coordinate direction.
  <g                                id="woodlandboundary"
style="&woodlandboundarystyle;">
      <path                           id="3139900173033"
onmouseover="showData('woodlandboundary','woodlandbound
ary')"  onmouseout="emptyData('woodlandboundary')"  d="M
314810 -171399 L 314810 -171490 L 314830 -171660 L
....
....
314480  -171820  L  314490  -171570  L  314490  -171340  L
314736 -171215 L 314810 -171300  z"/>
...
</g>
  // A Woodlandboundary is displayed  as a white area.
Create mouse actions on the object.
  ...
```

Fig. 5. Fragment of SVG code created by the server.

<map></map> tag is the ontology tree root, it should have one or more children which are defined by the <node></node> tag. The <node>tag has four children: name, parent, synonym and related. Each of the four children is an atomic element. Each node must have a name and a parent. It does not matter if the node has synonym words or related words.

Each Maplet represents an instance of a concept that can be mapped to a node in the OR. A search message from the user will be mapped to a node in the OR and all the Maplets mapping to the same node and a set of its immediate “relatives” will respond to the user's request. For example, if the user searches "transportation" information, then all the Maplets related in the ontology like motorways, railwaystations, railways etc. will display. The OR provides the potential to allow the UIA to perform imprecise matches between the user's query terms and the information in the information system. Techniques for performing such matches and to rank their results have been described in the context of geographical information in [25].

4.3 Ontology editor

The main goal of the ontology editor (OE) is to provide a GUI to create a geographical information related ontology described in XML. Based on the ontology, the User Interface Agent can quickly capture a user’s interactive intention from their inputs. The OE has been implemented in Java 1.2 using Java Swing components. It can run on Windows platforms and Sun Solaris. Figure 6 is a snapshot of the OE.

The following is a summary of the OE functions:

- Export XML/Ontology: The Export XML option is used to save the ontology created by the user as an XML file based on the DTD definition.
- Insert/Node: The Insert option is used to insert a new node into the ontology tree. When the user inserts a new node, he should first type in a new word in the text field of New Node and then select its parent node from the Parent Node combox. After that, he should type in its synonym words and semantic related words in the related text areas. After the user finishes the new node’s inputs, parent node, synonym and related words, he can select the Insert option to insert the new node into the ontology tree.
- Delete/Node: The Delete option is used to delete a node from the ontology tree. The node must be a leaf, otherwise it can not be deleted. When a node is deleted, its synonym words and related words are also deleted (this does not affect other references to the same words).
- Modify/Node: The Modify option is used to modify a node in the ontology tree. The user can select a node from The Existing Nodes combox and modify its name, its parent node and its synonym words and related words.
- New/DTD: The New option is used to input a new DTD file. At this stage, the OE only uses its pre-defined DTD file. The DTD file definition is given below.
- Load/DTD: The Load option is used to load an existing DTD file into the OE.
- Save/DTD: The Save option is used to save the DTD file.

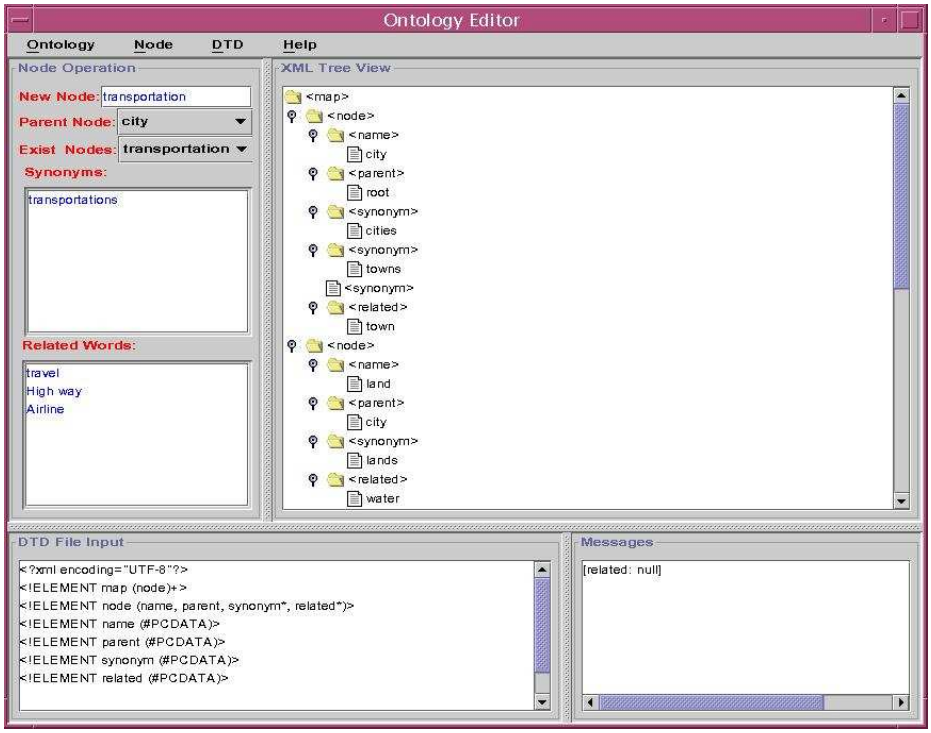

Fig. 6. A snapshot of the OE referring to a very limited ontology

5 An Agent-based Method for Graphic Conflict Resolution

Graphical conflict resolution is an important issue in presenting retrieved map information from the system. Here we describe the main characteristics of a graphical conflict resolution technique based on the principle of communicating agents. The work has been carried out with a view to incorporation of such techniques within the MAPBOT multi-agent map information retrieval system. We are concerned with graphical conflicts caused by map symbols being located too close to each other to be clearly discernable. Primary methods for resolving such conflict are displacement, deletion and amalgamation of the map symbols. In this study we focus on the operation of displacement, since its successful operation will result in the least information loss on the map.

The problem of on-line graphic conflict resolution by displacement can be divided into the following sub-problems:

- Proximity search procedure to detect conflicts;

- Strategy to decide the direction and distance of movement of objects in order to solve conflicts;
- Strategy of process flow control to manage interactions among objects.

Proximity search can be implemented efficiently by means of spatial indexing methods or through the use of a suitable data structure such as the constrained Delaunay triangulation (CDT). It constitutes however a major overhead in the process.

Among previous conflict resolutions methods, simulated annealing [26, 27] moves objects in conflict randomly on the basis of a set of pre-computed positions for each object. A pre-computed CDT is used for proximity search. Those moves which cause inversion of triangles, and hence may result in conflict, are rejected. Although this method is very efficient in identifying potential moves (as they are pre-computed), it generates a very large number of moves during the process (typically several orders of magnitude more than the number of objects in the dataset) and consequently a huge number of proximity search operations have to be carried out. In addition, possible new positions are limited so some potentially good positions may be missed. The method of [28] is based on vector algebra to calculate iteratively a movement vector derived from the conflict vectors applying to an object under conflict. This method generates many fewer moves than simulated annealing, but proximity search operations are still a major overhead of the procedure. Alternative approaches to displacement employ techniques, such as least squares [29] and finite element analysis [30], appear promising and have been applied to lines as well as polygons, but timing performance reports do not indicate their superiority. In an attempt to provide consistency in experimental comparison, we confine ourselves here to comparison with the former two methods.

Both of the methods considered move one object each time. After an object is moved to a new position, the status of the object as well as its neighbours has to be re-checked to find out distances and directions to neighbours and to detect conflicts. If only one object is moved each time, potentially there can be a lot of duplicated status checks. Alternatively, if several associated objects can be moved in a single step and status checks are carried out on the set of objects, i.e. the union of these moved objects and their neighbours, such duplications can be avoided. In addition, the number of overall moves may also be reduced as several conflicts may be solved at one time.

5.1 An agent-based architecture for conflict resolution

In order to move several objects in a single step without causing new conflicts, a mechanism is needed for objects to negotiate with each other to decide whether a move is allowed as well as the directions and distances of movements.

To achieve the above goals, we designed an agent architecture with two levels. At the top level, a house-keeping agent controls the process flow and monitors the overall condition of optimisation (number of remaining conflicts or process time elapsed); at the lower level, each map feature has a map agent with a “sensor” to check its environment, a “communicator” to send requests or responses to other

objects, a "solution-finder" to decide the contents of requests or responses as well as the ability to "move" itself to a new position.

5.2 The behaviour of map agents

The interaction among map agents can be described as a two-way negotiation:

- Making a request and finding a solution: when a map object O_0 is in conflict with another object O_1, if there is enough free space for it to move away from O_1 (i.e. along direction O_1O_0), the conflict is solved. Otherwise, its agent will send a message to O_1 and request O_1 to move away (i.e. along direction O_0O_1). If this request is implemented successfully, the conflict is solved. Alternatively, it may send a message to an opposite neighbouring object O_2 (i.e. O_0 is between O_1 and O_2) to move away to make some free space for it. If this request is successful, O_0 can then move away from O_1 and the conflict is also solved. The distance and direction in the requests are decided by the nature of the conflict, based on an analysis of local conflict vectors.
- Responding to a request: when an object O_1 receives a request from O_0, if there is enough free space in the requested direction, O_1 will make the move and send back a successful response to O_0 (with details of distance and direction of the move made). Otherwise, O_1 will send a new request to its conflicting neighbour O_3 in the requested direction and wait for O_3's response before responding to O_0. Note that this process may be carried out recursively and a chain of objects could be involved (and may be moved in a single step). Also, if a request can not be fully satisfied, a partial move may be made.

5.3 Implementation and test dataset

At present the architecture described above is implemented in C++.

For testing purposes we used an IGN-BDTopo dataset (Figure 7), which has been used in the previous conflict resolution research of [27 , 28]. The original dataset (dataset 1) contains 367 objects (46 linear objects as non-moveable region boundaries and 321 polygonal objects as moveable buildings). A simplified version (dataset 2) of the dataset (with one linear object removed and a building object simplified) is also used.

The conditions for conflict resolution are the same as in the previous studies, i.e.

- Building-Building distance tolerance 7.5
- Building-Boundary distance tolerance: 7.5
- Maximum displacement from original position constraint (MDC): 7.5

The two tolerances are minimum separation distances between two areal objects (buildings) and between an areal object and a linear object (boundary polyline). Initially there are 193 (dataset 1) / 191 (dataset 2) buildings in conflict.

5.4 Results

The results of the new method shown here in Tables 1 and 2 are averages of 10 runs. In Table 1, the row labelled "Conflict Aggregation" refers to results from the method of [28]. The row labelled "Simulated Annealing" provides results from [27]. The row labelled "Agent" reports the results from the new procedure. As they are quite stable, standard deviation is not shown. Figure 8 shows the result of applying the method to dataset 2, without using the minimum distance constraint. Figure 9 shows the dataset with reduced plot scale. Figure 10 shows detail while Figure 11 illustrates the results of applying the method without the minimum distance constraint to the same area. Figure 12 shows some detail of a part of dataset 2 with MDC constraint, while Figure 13 shows the results of applying the method without the constraint applied to the same area. These results demonstrate that the method based on communicating agents produced superior results to the other methods, both with regard to reducing the number of remaining conflicts and with respect to execution time.

Method	Platform	OS	Dataset	Exec. Time (sec)	Moves	Remaining Conflicts
Conflict Aggregation	PII-233 64M	Win95	1	78	~400 (?)	38
Simulated Annealing	PIII-800	Linux	2	1.55	37283.8 (s.d.1864.7)	26.5 (s.d.2.8)
Agent	Mobile PIII-700	NT5	2	0.799	1035	2

Table 1. Comparison of three methods (with MDC criterion)

Platform	OS	Dataset	Exec. Time (sec)	Remaining Conflicts
P-133/24MB	Win98	1	4.844	0
PIII-600/128MB	NT4	1	0.723	0
Mobile PIII-700/128MB	NT5	1	0.805	0
		2	0.533	0

Table 2. Results of Agent method on different platforms (WITHOUT MDC criterion)

An on-line demonstration of the agent procedure has been set up at:
http://www.cs.cf.ac.uk/user/S.Zhou/MSDBDemo/

Dataset 1 is used in this demonstration, while MDC is not applied.

Fig. 7. Dataset 2 - Original (solid objects are *NOT* in conflict)

Fig. 8. Dataset 2 - Conflict solved (without applying MDC)

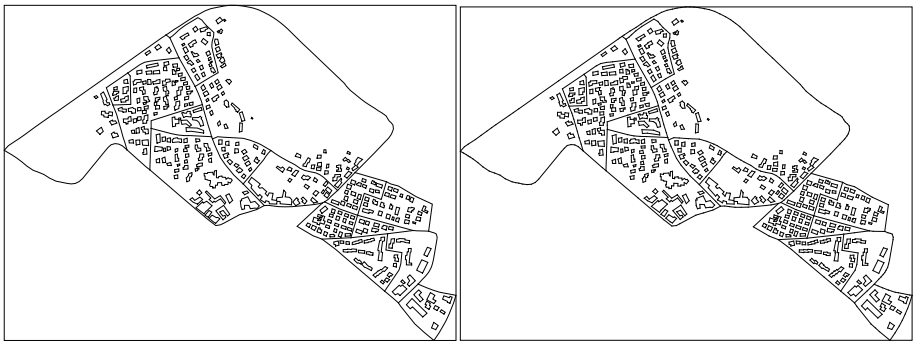

Fig. 9. left: original; right: conflicts removed without applying MDC.

(Plotting scale as 7.5m = 0.3mm on paper)

Fig. 10. Detail 1 (dataset2, with MDC, original position in dash-line)

Fig. 11. Detail 1 (dataset 2, without MDC, original position in dash-line)

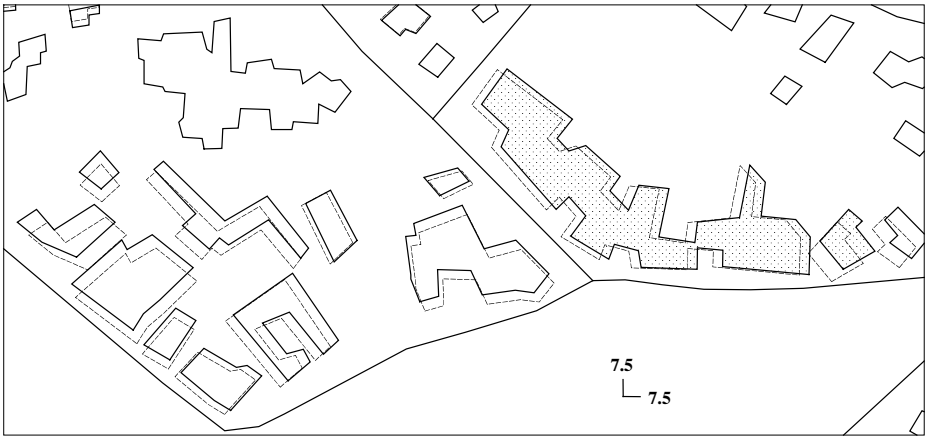

Fig. 12. Detail 2 (dataset 2, two objects remaining in conflict due to MDC)

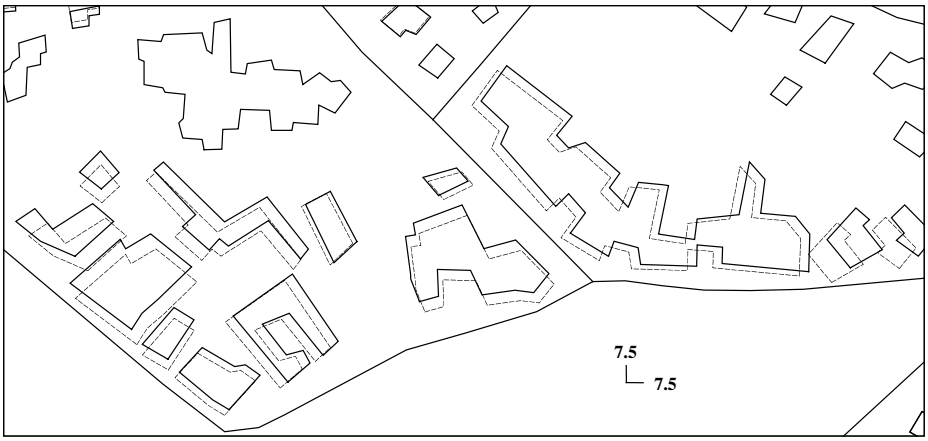

Fig. 13. Detail 2 (dataset 2, without MDC, all conflicts are solved)

5.5 Discussion

The type of graphic conflicts handled by the procedure described above can be regarded as a special case of the general problem of polygon separation which is NP-complete[33]. At present, the boundary polylines are not changeable during the process, which implies each plot (group of areal objects surrounding by a set of boundary polyines) in the dataset is independent of other plots and the total process time will not exceed the time spent on the "hardest" plot multiplied by the number of plots in the dataset, regardless of the total size of the dataset. Secondly, due to the presence of a fixed boundary, it is not always possible to find a solution to all conflicts (even without applying the MDC constraint). In our current implementation, we have set up several conditions (time elapsed, number of remaining conflicts, number of failed attempts, etc.) which are used in combination and tested periodically to decide when the program should be terminated if a feasible solution can not be found within a preset time limit.

It is worth repeating here that this algorithm primarily aims at on-line real time applications and therefore performance is our major concern. Other generalisation operators/algorithms may be integrated to improve the quality of the result only if the main algorithm is efficient enough, so that there is CPU time to spare within the performance threshold.

6. Future Work

Most web based map interfaces suffer from poor legibility of symbols and text, unnecessary user actions and inadequate adaptation to user interests. MAPBOT, an active map system using software agent technologies is presented to solve these problems. A system prototype has been implemented in Java to test the efficiency and

effectiveness of the active map. Based on the ontology repository, the system allows the user to perform some imprecise searches and the map interface changes dynamically to adapt to user interests. The visualisation system employs SVG to provide a high quality web map. The issue of maintaining a legible map display is a challenging one that requires a range of map generalisation functions. In this paper we have shown that agent approaches, based on local communication between neighbouring map features has the potential to solve graphical conflicts due to features becoming too close. The approach presented here has been found to have superior performance to iterative improvement procedures that examine many more potential map states using global control procedures.

The prototype has shown some promising results for the final goal of the MAPBOT. However, the project is ongoing and there are various areas in which further work must be expended. Examples of these include:

- In moving to a more fully-fledged agent environment the question arises as to the granularity of agency. At present each map feature is an independent agent, but this could result in very high level of communication. An alternative may be to allocate agents to the feature class level, and give them control over their respective map features.
- Add more functionalities to the UIA. Currently it only supports keyword search. Sentence recognition should be supported in the future based on natural language processing.
- Integrate the OE with an existing lexical system like WordNet [31] to allow the system developer to construct a more complete OR for the system.
- Allow Maplets to communicate with information agents outside the system to retrieve more information for the user.
- When a user submits a valid search, the Servlet creates the SVG based map file which is then downloaded to the client side to update the display. Since the map file size is big, this can introduce time delays. Consideration should be given to compressing the file.
- The MAPBOT system should be associated with a Multiscale Spatial Database [32] to provide efficient access to geographical map data at multiple levels of detail. The Maplets will then implement graphic generalisation operations on the retrieved data.

Reference

1. Stan Franklin and Art Graesser (1997). Is it an agent, or just a program? A taxonomy for autonomous agents. In *Third International Workshop on Agent Theories, Architectures and Languages*. Springer-Verlag, 21-35.
2. H. Nwana (1996). "Software Agent: An Overview". *Knowledge Engineering Review*, 11(3), 1-40.
3. C. Langton, M. Netson and B. Roger. The Swarm Simulation Systems: A Tool for studying Complex Systems. Santa Fe Institute, NM, http://www.swarm.org.
4. Batty M. and B. Jiang (1999). Multi-Agent Simulation: New Approaches to Exploring Space-Time Dynamics Within GIS, http:www.casa.ucl.ac.uk/multi_agent.pdf
5. N.R. Jennings, K. Sycara, et al (1998). A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*. 1(1): 7-36.

6. Ming-Hsiang Tsou and Barbara P. Buttenfield (1996). A Direct Manipulation Interface for Geographical Information Processing. *Proceedings 6th International Symposium on Spatial Data Handling*, Delft, The Netherlands: 13B.37 - 13B.47.
7. http://agent.ign.fr
8. D'Aloisi, D. and Giannini, V. The Info Agent: an Interface for Supporting Users in Intelligent Retrieval. *Fondazione Ugo Bordoni*, November 1995, Rome.
9. Jason I. Hong and James A. Landay, "A Context/CommunicationInformation Agent." In *Personal and Ubiquitous Computing: Special Issue on Situated Interaction and Context-Aware Computing* , 2001. 5(1): p. 78-81.
10. Balabanovic, M. and Shoham, Y. Learning Information RetrievalAgents: Experiments with Automated Web Browsing, *AAAI Spring Symposium on Information Gathering*, 1995, Stanford, CA.
11. Knoblock, C., Arens, Y. and Hsu, C. Cooperating Agents for Information Retrieval, Proceedings *of the Second International Conference on Cooperative Information Systems*, 1994, University of Toronto Press.
12. Lieberman, H. From Human-Computer Interface to Human-Computer Collaboration, CHI Research Symposium, 1995.
13. Resnick, P., Zeckhauser, R., & Avery, C. *Electronic brokers: Examples and policyimplications,* Twenty-Second Annual Telecommunications Policy Research Conference, 1994.
14. Rich, C. and Sidner, C. Adding a Collaborative Agent to Graphical User Interfaces, *ACM Symposium on User Interface Software Technology*, 1996.
15. Guarino, N., Masolo, C. and Vetere, G. OntoSeek: Using Large Linguistic Ontologies for Accessing On-Line Yellow Pages and Product Catalogs*, LABSED-CNR Technical Report 01/99*, Padova, Italy.
16. The InfoSleuth Agent System. See: http://www.mcc.com/projects/infosleuth/architecture/index.html
17. RETSINA, Robotics Institute, Carnegie Mellon University, 1998.
18. Chen, L. and Sycara, K. WebMate: A Personal Agent for Browsing and Searching, *Proceedings of the 2nd International Conference on Autonomous Agents and Multi-Agent Systems*, Minneapolis, 1998.
19. Borgo, S. and Guarino, N. An Ontology Theory of Physical Objects, *Proceedings of 1997 Workshop on Qualitative Reasoning* (QR 97), Cortona, Italy.
20. O'BRIEN, Paul, Paul. OSCA: An Ontology Based Sales Consultant's Assistant. Proceedings of ACIS'99.
21. http://www.w3.org/TR/SVG
22. P. Maes (1994). Modeling Adaptive Autonomous Agents, Artificial Life, 1:135-162.
23. http://www.adobe.com/svg
24. Alberts, L.K. (1993).YMIR: an Ontology for Engineering Design. University of Twente.
25. Jones C.B., H. Alani and D. Tudhope (2001). "Geographical information retrieval with ontologies of place", D. Montello (ed), *Spatial Information Theory Foundations of Geographic Information Science*, COSIT 2001, Lecture Notes in Computer Science 2205, Springer, 323-335.
26. Ware J.M. and C.B. Jones (1998) "Conflict Reduction in Map Generalisation Using Iterative Improvement", *GeoInformatica* 2(4), 383-407.
27. Ware J.M., C.B. Jones and N. Thomas (2001). "Map generalization, object displacement and simulated annealing: two techniques for execution time improvement", in D.B. Kidner and G. Higgs (eds), *Proceedings GIS Research UK GISRUK 2001*, University of Glamorgan, 36-38.

28. Lonergan, M.E. and C.B. Jones (2001). “An iterative displacement method for conflict resolution in map generalization”, *Algorithmica*, 30, 287-301.
29. Harrie, L. E. (1999). “The constraint method for solving spatial conflicts in cartographic generalization.” *Cartography and Geographic Information Science* 26(1), 55-69.
30. Hojholt, P. (1998). "Solving local and global space conflicts in map generalisation using a finite element method adapted from structural mechanics". *8th International Symposium on Spatial Data Handling,* Vancouver, International Geographical Union, 679-689.
31. http://www.cogsci.princeton.edu/~wn
32. Zhou, S. and C.B. Jones (2001). Design and Implementation of Multi-Scale Databases. *7th International Symposium on Spatial and Temporal Databases SSTD01*, Lecture Notes in Computer Science 2121, 365-84.
33. Li, ZY and V. Milenkovic, (1995). Compaction and Separation Algorithms for Non-Convex Polygons and Their Applications. European Journal of Operations Research, 84, 539-561.