DART: The Distributed Agent-Based Retrieval Toolkit

MANUELA ANGIONI, ROBERTO DEMONTIS, MASSIMO DERIU,
EMANUELA DE VITA, CRISTIAN LAI, IVAN MARCIALIS,
ANTONIO PINTUS, ANDREA PIRAS, ALESSANDRO SORO, FRANCO TUVERI
CRS4 - Center for Advanced Studies, Research and Development in Sardinia
Polaris, Edificio 1, 09010 Pula (CA)
ITALY
{angioni, demontis, mderiu, emy, clai, ciano, pintux, piras, asoro, tuveri}@crs4.it http://www.crs4.it

Abstract: The technology of search engines is evolving from indexing and classification of web resources based on keywords to more sophisticated techniques which take into account the meaning and the context of textual information and usage. Replying to query, commercial search engines face the user requests with a large amount of results, mostly useless or only partially related to the request; the subsequent refinement, operated downloading and examining as much pages as possible and simply ignoring whatever stays behind the first few pages, is left up to the user. Furthermore, architectures based on centralized indexes, allow commercial search engines to control the advertisement of online information, in contrast to P2P architectures that focus the attention on user requirements involving the end user in search engine maintenance and operation. To address such wishes, new search engines should focus on three key aspects: semantics, geo-referencing, collaboration/distribution. Semantic analysis lets to increase the results relevance. The geo-referencing of catalogued resources allows contextualisation based on user position. Collaboration distributes storage, processing, and *trust* on a world-wide network of nodes running on users' computers, getting rid of bottlenecks and central points of failures. In this paper, we describe the studies, the concepts and the solutions developed in the DART project to introduce these three key features in a novel search engine architecture.

Key-Words: Search Engine, Geo-Reference, Semantics, NLP, RDHT, DART, 3D-UI, Community, P2P.

1 Introduction

The web is the largest and most untidy data source people can use. Search engines help people to find information, although, often, not exactly the information they need. While indexing and querying such a large amount of resources is within the capabilities of commercial search engines, the ability to filter, select and separate what is relevant to the user.

To date, users know that most of the results they get in reply to a query, are useless or only partially related to their requests, and are resigned to choose by hand, usually among the first few results, and to ignore all the rest. While query results can be improved refining the request to make it more precise, a real efficacy increase can only be achieved introducing the *context* and the *meaning* as fundamental concepts in query resolution and formulation. A further weakness is the analyse of the deep web: the Web not accessible through the search engines [1].

The new generation of search engines requires advanced features and new architectures to find the virtual web objects (i.e. HTML pages, images, videos, and any kind of files) and especially concrete objects and services, on which search engines have to focus their next effort. The users want information about real object characteristics, available products in a supermarket or a shop, a parking space close to home, the nearest restaurant to their current position, the post office with shorter waiting time, and so on. These are categories of information required to support nomadic people, to satisfy the human desire of knowledge and to improve the quality of life.

To address such wishes, the new search engine should focus on three key aspects:

- semantics,
- geo-referencing,
- collaboration / distribution.

Semantic analyses let automatic systems to create a structure able to give the right meaning to groups of words according to the contiguous sentences and solving misunderstandings related to thesaurus and slang expressions, improving web information retrieval, knowledge management and enterprise application integration.

Thanks to the management of geo-referenced data, the user will submit questions related to the position specified by latitude-longitude coordinates or by place names. The search engines will automatically process reverse geo-coding either during the page parsing and user query processing while mobile objects and people will spontaneously notify their position.

Collaboration is the best solution to find the invisible web and to distribute the processing power required to scan and catalogue its pieces of information. It is a collaboration between remote peers but specially it is a

user collaboration. In fact they may provide their help submitting directly new resource and offering storage space and bandwidth of their Internet connection. In such way, there is no central control system, avoiding bottlenecks and central points of failures, and the ranking system will be public.

In this paper we expose the studies, the concepts and the solutions developed in the DART project to introduce semantics, geo-referencing and collaboration features in a search engine. Section 2 provides a general overview of its architecture and the main modules. The following sections, from 3 to 8, describe each module exposing the indexing and storing solutions, the semantic processing, the resource submitting and the HMI proposed to user.

2 The DART Project

DART stands for Distributed Agent-based Retrieval Toolkit and it is a research project aimed at studying, developing and testing patterns and integrated tools to achieve a semantic, distributed geo-sensible search engine, defined focusing on users requirements and giving them the possibility to be involved in search engine activities.

The spreading of mobile devices and wireless networks makes possible the definition of a system able to support mobile users and manage information related to their position where submitted queries are automatically enriched using their profile, the device profile and the position context. The adoption of a semantic approach in resource cataloguing and in query resolution allows results filtering increasing the overall response quality, while the computation is distributed on a network of nodes directly managed by the user community.

The wide range of technological aspects of such search engine suggests to design the DART node using a modular approach (see Fig. 1).

A set of Data Providers collect information parsing web pages, accessing Peer-to-Peer (P2P) networks, reading Universal Description, Discovery and Integration (UDDI) registries, grabbing Electronic Program Guide (EPG) messages. Such information is processed by the Semantic Module that delivers the utilities of distributed indexing and storing layer to add resources on the catalogue of the node network. Furthermore, sensors and ad-hoc applications automatically and periodically push values to the DART system.

The Query Module receives requests submitted by users through web pages, web services, or ad-hoc applications embedded in portable devices. The queries are enriched with the user profile, the device profile and the position related information and forwarded to the semantic module to be addressed The search results are collected and filtered using again the user profile, the device profile and the context information and returned.

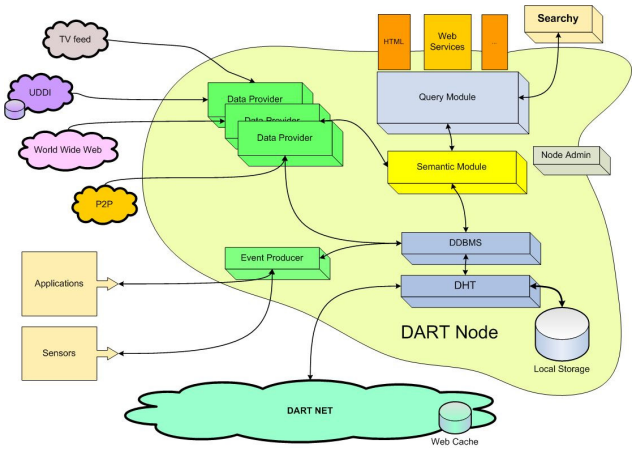

Fig. 1. Architecture of the DART Community Node.

An Autonomous Interface Agent (AIA), Searchy, runs as a web browser extension and provides advice during the browsing session. Finally, the DART Community Node includes an administration application in order to allow a basic parameterized configuration of the system, like starting or stopping the web crawling and specifying TCP, UDP, HTTP ports.

3 The DART Community Node

The DART Community Node sets up the main software entity for the DART project both for architectural aspects and prototypal points of view. It is composed of distinct interconnected software modules, each one providing a specific functionality to the system.

DART Community Nodes can be used and employed in several configurations, from closed nodes clusters to the scenario of a totally open web based community composed by users, as shown in Fig. 2. The last one is the configuration to let us to achieve the target of a node collaboration in indexing and storing crawled web information.

Users can join the DART community simply installing a community node on their computer, becoming capable to share computation resources with the community, contribute to the web crawling (as in [2]) and participate to a distributed storage keeping a portion of the information of the global index.

In its default configuration a DART Community Node is able to perform a network discovery step, in order to retrieve another community node and join to the community. The node is capable to collect data coming from Data Providers, for example a web crawler, and then forward specific types of collected data to the Semantic Module for semantic analysis and cataloguing. That module uses the DDBMS module to properly store processed data. Other data types, such as web links,

extracted from web pages, are directly examined and stored in the DDBMS, wrapped with a special data descriptor which specifies metadata and useful information for distributing the crawling tasks.
Using the DART Community Node, users contribute to the realization of a complete, updated, and *public* web snapshot where they are able to explore and retrieve information and resources contained via a semantic, personalized and flexible query resolution.

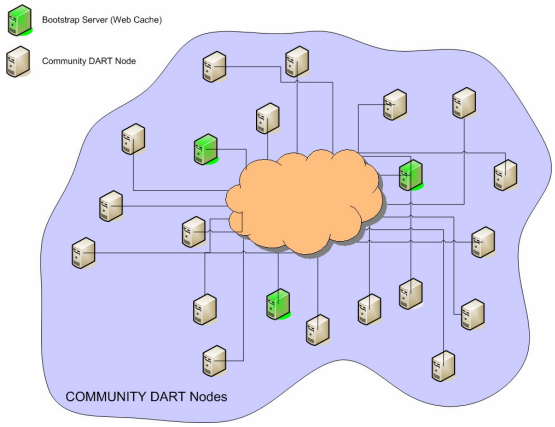

Fig. 2. A full open community of DART Community Nodes.

4 DHT and Distributed DBMS

DART users are supposed to contribute to the system in terms of storage and CPU cycles, but also sharing information as occurring in P2P applications. As a backend to DART, an efficient, robust and scalable distributed file system is required and Distributed Hash Tables (DHTs) over a P2P overlay have extensively proven to meet these requirements.
The most important features related to DHTs can in fact be summarized as follow:

- efficiency and scalability, the number of messages exchanged to route a query to its destination is O(log(N)), where N is the total number of nodes;
- no maintenance, no administrative operations are required, no central authority or complex process is required to maintain, balance or fix the distributed data structure;
- simplicity, the algorithms behind DHTs are relatively simple to understand and implement;
- robustness, the ability to survive massive failures is a key aspect when deploying largely distributed applications, such as file sharing applications.

DHTs can store and retrieve efficiently a huge amount of information, but queries require an exact knowledge of the resource ID (the key). This excludes many applications, in which the key is known only approximately, i.e. a geographical position, or is known to fall within a range, i.e. a time interval.
The DART project defines a DART Network Overlay, called DDBMS, whose goal is to support a wide variety of distributed applications, by providing a flexible, efficient, and robust distributed file system, capable of range queries. We call this file system RDHT (Range capable Distributed Hash Table) [3].
The RDHT file system is inspired by the skip lists [4], though in RDHT we lose the concept of different levels of pointers, in exchange for a self-organized backbone. The storage of index information, as well as any other operation, is built on top of the Kademlia [5] protocol. Index information is maintained in terms of pointers that link each element stored to its neighbours. Pointers do not get deleted in consequence of new inserts, as they do not link each element to its next, but simply represent a linear ordering relation. Storing a pointer from an item A to an item B means: B is greater than A. If subsequently a new element C is inserted in the RDHT that falls within A and B, while new pointers get stored from a A to C and from C to B, the old pointer from A to B remain valid, as the relative position of A and B is unchanged. Note that old pointers get gradually stretched to form a backbone that is used in fast lookup operations. Thanks to such organized pointers, it is always possible to reconstruct the complete list, even if for each value several neighbours are known: the element next to a given element E can be simply reached choosing the shortest possible pointer that has E as its base.
Also note that range queries only make sense if the items stored can be ordered with respect to one or more attributes. The RDHT stores only integer values. The transformation from Objects attributes to integer values is application specific: for certain applications a lexicographic order, resulting in a list, can be applied. In geographic applications, for instance, linearization is often used to represent n-dimensional coordinates. For example geographical coordinates can be linearized using a z-curve [6], i.e. interleaving the bits of the X and Y (or longitude and latitude) coordinates of the spots to obtain a single integer value. Items' coordinates are represented as an integer value, that is a point in the z-curve, and the limits of a query for a rectangular region are translated to a linear interval of the z-curve itself. It simplifies store and retrieve operations.
The RDHT exposes four primitive operations: lookup, nearest, insert and range. Low level primitives, relative to the DHT operations, like join, ping, etc. are implemented in the Kademlia network overlay, and will not be described here.
Lookup operation discovers an item stored in the data structure. It starts from a known value and executes several lookup operations on the underlying DHT, in

order to fetch index information. Pointers from the base to the target value are fetched in a recursive algorithm that executes at each step the longest possible jump that do not overshoots the target element. The lookup operation fails if the shortest possible jump from every known base overshoots the target value.
Nearest primitive simply consist of a lookup operations that, instead of failing if the given target is not found, simply returns the immediate successor and predecessor of the target value, that is: if the base is smaller than the target and the next to the base is greater, then these two elements are those nearest to the target.
Insert operations execute first a lookup to spot the nearest elements A and B to the target in the RDHT; only if the lookup fails the new element gets stored and pointers from A to target and from target to B are stored in the RDHT.
Range queries are banally executed following the shortest pointers from the lower bound to the upper bound of the query. Alternatively a range query can be executed searching for the items nearest to the median(lb, ub), and then repeating this operation recursively over the two subintervals until no new element is discovered or a given grain is reached. This second strategy can be easily parallelized.
Note that there is no remove primitive, once stored an item cannot be deleted. This implies that malicious removal of data is not possible.

5 Collecting data

We have identified two ways which DART system can retrieve information. In the first one, a module specialized in a data category reads and pre-processes data. If it requires a semantic analysis, data are forwarded to Semantic Module, otherwise it is given to DDBMS Module to be stored on RDHT.
The second approach is the submission of event disclosed by a sensor, an application or users.

5.1 The Data Providers

The Data Provider is a family of software modules able to retrieve resources available from the web, from P2P network or collected by iTV, and to process them in order to retrieve all relevant information to be sent to Semantic Module for indexing or directly storing on the DDBMS.
Each node contributes to crawl according to the hardware configuration and user's preferences. The crawling is distributed because every node, according to the optimization policy, can decide to assign the crawling of a resource to another node. The crawling functionality has to be switched on or off according to the user preferences. If it is switched off, the node does not participate to the crawling but can still offer the support for the other functionalities.
Some P2P bridges retrieve information from the existing resources on the other P2P networks (i.e. Gnutella, eDonkey2K, etc).
To collect EPG information and forwarding it to DART system, a tailored grabber may be installed on STBs (Set Top Boxes) and media centers.
Regarding to Web Services (WSs), a module inquiries UDDI registries in order to discover WSs. A special WS data provider is the GIS Data Provider. It indexes metadata and WSs described in a Catalogue Web Service [7]. Such data includes maps described in image or XML format and generated using Web Map Service [8] and Web Feature Service [9].

5.2 The Event Producer

The Event producer is a family of software modules able to produce dynamic information stored in DART system. To distinguish those from the other information we call them events.
The events can be generated by:

- sensors (RFID, etc.) installed on the device that hosts the DART node;
- peripherals installed on the device that hosts the DART node;
- local applications installed on the device that hosts the DART node;
- remote applications which require the DART node to save information;
- the user who wants to notify something to the system.

6 Semantic content management

Web development makes available documents and services in great number whose categorization, presentation and availability are making the information retrieval a very relevant topic.
In DART, we introduce semantic techniques, like ontologies and Natural Language Processing (NLP) tools, in order to exceed search engines limits. The Semantic Module analyses the collection of words, the relative weight, the link between them and the way they are semantically connected and processes the queries with reference to their semantic. These techniques allow to process queries formulated in Natural Language and find documents with the same semantic concept, ensuring in this manner a good level of relevance of the data provided to the user. Fig.3 depicts the Semantic Module parts and correlations with Data Providers ad DDBMS module.

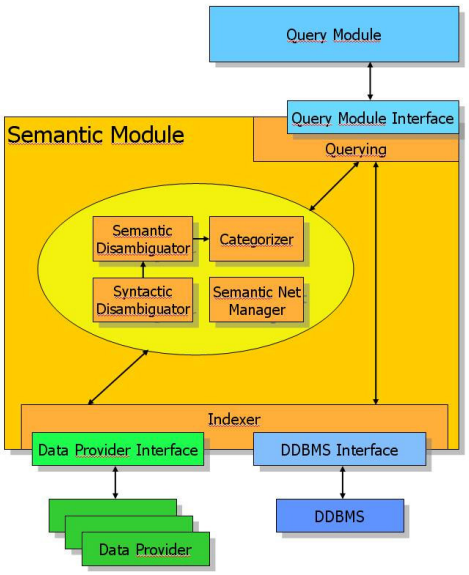

Fig. 3. The Semantic Module.

This module exchanges information with Data Providers, asking for resources, processes a semantic analysis on textual resources, by means of NLP techniques in addition to interpretation based on specific ontologies of annotated documents,. It uses the interface with the DDBMS Module to create a semantic indexing of ontology-based annotated resources and categories. Moreover it makes a semantic analysis and a categorization of queries received by the Query Module, resolves them and return results to the Query Module.
Its components are:

- the Syntactic Disambiguator is essentially composed by a syntactic analyzer, that uses the Link Grammar parser [10], a highly lexical, context-free formalism, to identify the syntactical structure of phrases and sentences, in order to resolve the roles of terms ambiguity present in natural languages;
- the Semantic Disambiguator analyses each phrase identifying roles, senses of terms and their semantic relations in order to extract "part of speech" information, the synonymy and hypernymy relations from WordNet [11] and to change the representation from words contained in a document to a density function based on the synonyms and hypernyms frequency [12];
- the Categorizer manages resources and queries classification by means of text categorization techniques;
- the Semantic Net Manager manages the building of the Semantic Net, composed by a set of topics linked through their senses, and uses it to enrich results with topics semantically related to queries submitted by users.

6.1 Semantic querying

In searching phase, the Semantic Module performs searches by keywords, by queries expressed in natural language and by categories, enriched by related topics given by the Semantic Net Manager.
The query, submitted through the Query Module Interface, is analyzed by the Semantic Module components obtaining a set of keys. Then, it is categorized and user feedback process starts to refine the range of the query, working on results threshold and categories lists. Undesired keys for the context are eliminated and a new list of categories updates the categories list previously given by the Semantic Module. Then results are returned to the Query Module.

6.2 Semantic Indexing

The Semantic Indexing is performed by a collaboration between the Semantic Module and Data Providers. The web resource is parsed by a specific Data Provider who identifies, into its content, structured (annotated) and unstructured (not-annotated) portions. The semantic indexing of a structured document is quite simple, by using the ontology or the structure descriptor. Otherwise a not annotated portion of a document needs a linguistic analysis of its content. The Data Provider indexes the structured portions of the document while the Semantic Module its unstructured parts.
In this paragraph we describe two examples of such different situations. The first one describes the semantic analysis of a not annotated document. In this case the semantic interpretation is performed by the Semantic Module. The text is split in phrases.
Some NLP tasks are executed for each phrase: extraction of "parts of speech" and syntactical relations by means of the Syntactic Disambiguator, the semantic analysis, the extraction of senses by means of the Semantic Disambiguator and the extraction of categories by means of the Categorizer. Disambiguation plays an important role in semantic index realization because reducing the false positive in the search phase. After the semantic analysis of each phrase, the Semantic Module catalogues documents applying text categorization techniques and writes all acquired information in the DDBMS.
The second situation is related to the semantic indexing of annotated text, performed on dynamic XML documents generated by WSs. They are not considered by search engines because their indexing is possible only if functionalities, contents and concerning ontologies of a WS are known a priori. In the DART project, the dynamic content of specified classes of WS is indexed

by a specific Data Provider that knows their functionalities and the representative ontology. However dynamic content generated by WS can contain portions of not annotated text that need NLP instruments to be interpreted. Those data can be accessed by means of a sense or a category key, based on their description, or by means of a geographical key. To manage this task the text is submitted to the Semantic Module.

7 Submitting queries

The Query Module has the responsibility to collect user queries, process them and perform a forwarding to the Semantic Module. It is composed by a set of components.

The Query Manager builds a user interface (UI) dependent on: the device used by users to query the system, the particular user context and the searched resourced type. UIs are developed using HTML, VoiceXML, X3D and XUL (XML User Interface Language) whereas B2B interfaces are implemented through SOAP (Simple Object Access Protocol) and REST (Representational State Transfer) based WSs.

The Device Manager is delegated to provide information useful for an optimal rendering of graphical user interfaces processing information collected by the virtual assistant installed on user device (see section 8.2). Furthermore, it has the responsibility to use device features for enriching user queries and to support results filtering in collaboration with the Result Collector module.

The User Manager provides user information details to the Query Manager, in query enrichment task, and to the Result Collector, for results filtering.

The Result Collector receives the collection of results from the Semantic Module. It processes a more accurate filtering of the results thanks to data provided by the User Manager.

Thanks to the Event Manager, the user can configure a DART Node in order to produce an alert when a particular event happens. The Event Manager is responsible of the user subscription for a particular type of event. The subscription is wrapped in a query and stored on the DHT. When the subscription is satisfied by an event, a notification is automatically forwarded to subscribers.

8 HMI in DART

The Human Machine Interaction (HMI) is an open and important question in ICT research. In scientific literature, several works study the design for more intuitive, adaptable and accessible user interfaces. One of the most interesting arguments is how the user can submit a question to the machine and how the machine provides the answer. In the web, such tasks are delegated to the search engines. Their scientific research and technology evolution has been focused firstly into the development of new algorithms for query processing, indexing of resources, and data caching. The most used search engines have homologated the procedure to submit a query and to present the results to the user. Except rare case, the results are showed like a collection of HTML pages that contains a list of resources with a brief description, if the total number of results exceeds the number of results that can be showed in a single page, new pages are generated. This type of web UI is a "de-facto" standard, specially because it's easy to use and simple to manage.

In the DART project, we aims to offer to the users a different approach, focusing our attention around three different points: to have an exhaustive user profile; to implement an intelligent virtual assistant and to show a 3D visualization of query responses.

8.1 User Profile in DART

The user profile collects all user data which the system uses to execute the login, to screen query results, to support the user in normal web navigation and to perform independent actions. These data items are stored in a server in order to be available for any device. They are downloaded into the client at the beginning of the session and remotely saved every time they are modifies by the user or the system. Format and amount of data can change in relation to the used device.

The DART User Profile ranges from personal information to friends, houses, jobs and hobbies. The virtual assistant detected data autonomously, including the login time and position, resources viewed (sites, pages, movies, WSs, etc.), the time spent on a page and a site, the pages added in browser bookmarks.

8.2 Searchy

An interesting research line is the study, the development and the implementation of a virtual assistant. It is an AIA that assists and interacts with the user during a browsing session. We have called it Searchy and it is an extension for Mozilla Firefox. Its tasks are to analyse user requests, if available data related to user position, downloaded pages and suggest related links, arguments, names or words. According to software agent features, Searchy is not an intrusive extension. It does not limit user interactions and it can be disabled anytime.

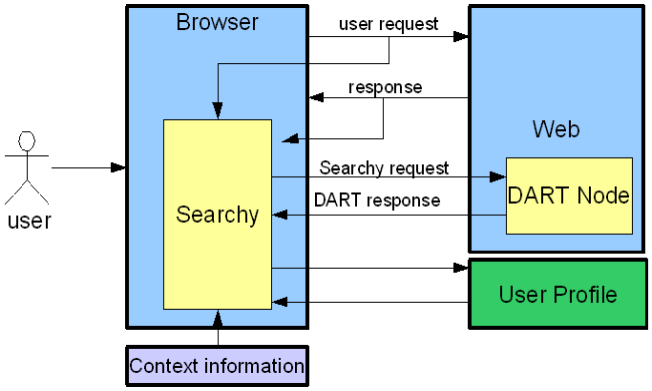

Fig. 4. Correlation between web browser, Searchy and DART Node.

The operative modality of Searchy can be divided in two different phases. The first one is absolutely transparent for the user who can carry out his work on the Web without knowing what Searchy is doing. In fact, every times the user, through the web browser, requests a new web page, Searchy intercepts HTTP request and analyzes the downloaded page. The first step of this analysis is to understand the page topics, this work is referred by Searchy to the Semantic Module which can extract topics and links of the page and combines the found subjects with a taxonomy of known topics. At the same time Searchy analyzes the user navigation session and the user profile (including the temporary context of the user). So, combining page topics, history and user profile, Searchy composes a query to submit to DART.
The second phase of the its processing is related to receive the DART response. The virtual assistant analyzes response, screens the list, removes banned or useless sites and performs the list ordering. Finally, Searchy shows the list in a side-bar of Mozilla Firefox.

8.3 3D User Interface for results.

In the DART project we explore the concepts of post-WIMP user interface in order to overcome the limits of XML-based UI, specially in terms of effectiveness and usability. These limits have driven the need of a new interface able to show results in a suitable, concise, and more effective way.
We have adopted a 3D visualization to allow the user to change the point of view, improving the perception and the understanding of contents [13]. In a 3D space, a user can easily understand the meaning of an object, simply rotating, shifting, and moving it. If the representation is suitable for the search context, the objects are easy to explore, and the related information are learned faster and better.
The 3D module provides an interactive representation of results designing 3D visual search interface characterized by concrete representations and simplicity [14]. It is an optional layer that could be used in base of the user preferences and on the context. According to the search context it provides a three-dimensional view, building an X3D [15] document that contains the most suitable scene. Moreover it could provide different layouts for different cases ([16] and [17]).
The choice of X3D as language to describe virtual world has been driven by its features, specially because it is a standard based on XML, and it is supported by a large community of users.

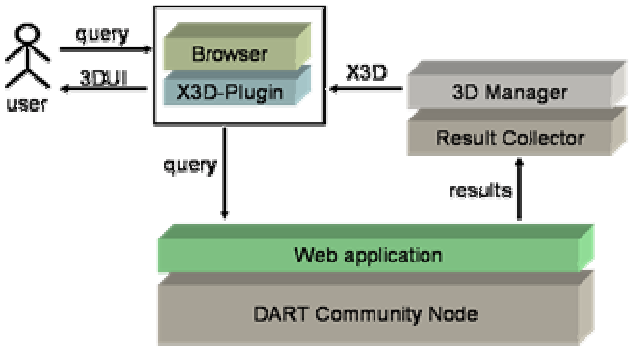

Fig. 5. The 3D-UI module.

The module parts are:

- a web application is the interface between the DART Node and the front-end of the system. Its tasks are to submit the query originated in the browser to the search engine, to forward results to the ResultCollector, and finally to encapsulate and to send the X3D document to the browser.
- ResultCollector, has the role to collect the list of results and process their arrangement according to file type and/or the relevance.
- 3D Manager receives the list of results and generates on the fly the X3D document that offers an interactive 3D scene with the search results. Subsequently it sends the document to the web application that makes it available to the user browser.

9 Related Work

Semantics aware search engine and frameworks based on ontologies are developed as a search technology for the semantic web (examples are [18], [19] and [20]).
The application of P2P paradigm to realize a distributed search engine architecture is targeted at getting rid of the typical problems of centralized systems, such as network overload, single point of failure, censorship, lack of scalability. Examples of community oriented architectures for geographic based services are exposed

in [21], [22], [23] and [24], while [25], [26] and [27] are examples of distributed search engines.

P2P systems have the problem to process range queries. Unstructured systems address it flooding queries to all peers in the network, thus requiring O(N) messages. P-Trees [28] and Prefix Hash Trees [29] are scalable solutions requiring to store a distributed indexing data structure in the P2P network itself, and use this to guide range queries. We propose the adoption of an overlay adding primitives to manage range queries on the Kademlia system.

Regarding HMI, in order to improve the DART usability, the studies focused on 3D UI for the visualization of results and the virtual assistant. Searchy combines several aspects described on various works: an advice system (like [30] and [31]), the automatic update of user preferences by the analyse of web-browsing behaviours ([32]) and the use of user profile to complete user requests ([33]).

10 Conclusions

In this paper, we have presented the DART project and the patterns and technologies exploited in the design of a distributed, semantic and context aware search engine. The goal of this research project is to provide users with a powerful toolkit for indexing online resources in a distributed and open database, managed by users themselves, and effectively querying this database with all the power and flexibility of natural language. DART includes tools to support personalization and management of the user profile, and is specifically designed to be accessed from virtually any device, adapting to the specific capability and the context of use. It aspires to overcome many limitations of current search engines, through state of the art technology in distributed systems, semantic web, and human machine interaction, Furthermore it is focused on the main goal of leveraging the birth of an online community of users, that share storage and computational power to the common objective of indexing and searching digital resources.

The DART project has been partially funded by the Italian Ministry of University and Scientific Research, contract grant number 11582.

References:

[1] M. K. Bergman, The Deep Web: Surfacing Hidden Value, *Journal of Electronic Publishing*, University of Michigan Press, Vol. 7, 2001.

[2] B. T. Loo, O. Cooper, S. Krishnamurthy, Distributed Web Crawling over DHTs, http://digitalassets.lib.berkeley.edu/techreports/ucb/text/CSD-04-1305.pdf.

[3] A. Soro, C Lai, Range-capable Distributed Hash Tables, in *Third International Workshop on Geographic Information Retrieval - GIR'06*, Seattle – USA, 2006.

[4] W. Pugh, Skip Lists: A probabilistic alternative to Balanced Trees, in *Workshop on Algorithms and Data Structures*, 1990.

[5] P. Maymounkov, D. Mazières. Kademlia: A peer-to-peer information system based on the xor metric, in *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, London - UK, Springer-Verlag, 2002, pp 53–65.

[6] H. V. Jagadish, Linear clustering of objects with multiple attributes, in *ACM SIGMOD International Conference on Management of Data (SIGMOD'90)*, 1990, pp. 332–342.

[7] Open Geospatial Consortium, "Catalogue Web Service", http://www.opengeospatial.org/standards/cat.

[8] Open Geospatial Consortium, "Web Map Service", http://www.opengeospatial.org/standards/wms.

[9] Open geospatial Consortium, "Web Feature Service", http://www.opengeospatial.org/standards/wfs.

[10] D. D. Sleator, D. Temperley, Parsing English with a Link Grammar, in *Third International Workshop on Parsing Technologies*, 1993.

[11] Wordnet, http://wordnet.princeton.edu.

[12] S. Scott, S. Matwin, Text Classification using WordNet Hypernyms, in *COLING/ACL Workshop on Usage of WordNet in Natural Language Processing Systems*, Montreal, 1998.

[13] J. Biström, Al Cogliati, K. Rouhiainen, Post-WIMP User Interface Model for 3D Web Applications, Helsinki University of Technology Telecommunications Software and Multimedia Laboratory.

[14] B. Houston, Z. Jacobson, A Simple 3D Visual Text Retrieval Interface, in *TRO-MP-050 - Multimedia Visualization of Massive Military Datasets. Workshop Proceedings*, 2002.

[15] Web 3D Consortium - Overnet. http://www.web3d.org.

[16] W. Wiza, K. Walczak, W. Cellary, AVE - Method for 3D Visualization of Search Results, in *3rd International Conference on Web Engineering ICWE*, Oviedo – Spain, Springer Verlag, 2003.

[17] N. Bonnel, A. Cotarmanac'h, A.Morin. Meaning Metaphor for Visualizing Search Results, in *International Conference on Information Visualisation*, IEEE Computer Society, 2005, pp. 467–472.

[18] U. Straccia, R. Troncy, Towards Distributed Information Retrieval in *Semantic Web, in 3rd*

European Semantic Web Conference (ESWC-06). Springer Verlag, 2006.

[19] L. Ding, T. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, J. Sachs, 2004. Swoogle: a search and metadata engine for the semantic web, in *Thirteenth ACM international Conference on information and Knowledge Management*, Washington – USA, ACM Press, 2004, pp. 652-659.

[20] C. Rocha, D. Schwabe, M. P. Aragao, A hybrid approach for searching in the semantic web, in *13th international Conference on World Wide Web*, New York - USA, ACM Press, 2004, pp. 374-383.

[21] S. Balram, S. Dragicevic, Collaborative Geographic Information Systems: Origins, Boundaries and Structures.

[22] D. Carboni, S. Sanna, P. Zanarini, GeoPix: Image Retrieval on the Geo Web, from Camera Click to Mouse Click, in *MobileHCI'06*, Helsinki - Finland, ACM Press, 2006

[23] J.H. Guan, S.G. Zhou, L.C. Wang, F.L. Bian, Peer to Peer Based GIS Web Services, in *XXth ISPRS Congress*, Istanbul – Turkey, 2004.

[24] A. M. MacEachren, G. Cai, R. Sharma, I. Rauschert, I. Brewer, L. Bolelli, B. Shaparenko, S. Fuhrmann, H. Wang, Enabling Collaborative Geoinformation Access and Decision-Making Through a Natural, Multimodal Interface. *International Journal of Geographical Information Science.*

[25] A. Singh, M. Srivatsa, L. Liu, T. Miller, Apoidea: A Decentralized Peer-to-Peer Architecture for Crawling the World Wide Web, in *SIGIR 2003 Workshop on Distributed Information Retrieval*, Lecture Notes in Computer Science, Volume 2924, 2003.

[26] Yacy, http://www.yacy.net/yacy.

[27] J. Callan, Distributed information retrieval, *Advances in Information Retrieval*, Kluwer Academic Publishers, 2000, pp. 127-150.

[28] A. Crainiceanu, et Al, Querying Peer-toPeer Networks Using P-Trees, in *WebDB Workshop*, 2004.

[29] S. Ramabhadran, S. Ratnasamy, J. Hellerstein, S. Shenker, Prefix Hash Tree - An Indexing Data Structure over Distributed Hash Tables, 2004.

[30] L. Chen, K. Sycara, WebMate: a personal agent for browsing and searching, in *Second international Conference on Autonomous Agents*, Minneapolis - United States, ACM Press, 1998, pp. 132-139.

[31] H. Lieberman. Letizia: An agent that assists web browsing, in *International Joint Conference of Artificial Intelligence*, Montreal - Canada, 1995.

[32] Y. Seo, B. Zhang, Learning user's preferences by analyzing Web-browsing behaviours, in *Fourth international Conference on Autonomous Agents*, Barcelona - Spain, ACM Press, 200, pp. 381-387.

[33] G. L. Somlo, A. E. Howe, Using web helper agent profiles in query generation, in *AAMAS '03: second international joint conference on Autonomous agents and multiagent systems*, ACM Press, 2003, pp. 812-818.