A Multi-Agent Platform Supporting Maintenance Companies on the Field

Andrea Passadore, Giorgio Pezzuto, *D'Appolonia S.p.A. Via San Nazaro 19, 16145 Genova, Italy*

Abstract—In this paper we present the European project named E-Support, aimed to the maintenance companies which work on the field, away from the central headquarters. The main goal of E-Support is to help the field engineers and technicians to access the knowledge base of the company. They will connect to remote servers by using mobile devices in order to get information about vendors, customers, plants, parts and download technical documents. The whole system will be implemented by a multi-agent platform running agents on mobile devices and server agents that provide the services. A particular emphasis will be placed on the contribution of D'Appolonia regarding the document retrieval system.

***Index Terms*—Enterprise, multi-agent, document, indexing, clustering, ontology.**

I. Introduction

E-SUPPORT is a collective European project which involves RTD (Research and Technology Development) companies, SMEs (Small and Medium Enterprises), and associations of maintenance companies. The project is addressed to these maintenance companies which send small teams of engineers and technicians at the customer's, in order to fix, upgrade or simply manage a plant. Often the personnel which works away from the headquarters needs information in real-time regarding the customers, vendors, parts to replace and any type of data useful during the everyday work activity. The aim of E-Support is to provide these data on the field, directly on the plant, connecting a mobile device to the remote server containing all the information owned by the company and the information offered by the associations (as regulations, norms, standards, etc.) that reunite maintenance companies. E-Support enables technicians to use every type of mobile device: mobile phones, smart phones, PDAs (Personal Digital Assistant), and notebooks. It is a task of the remote system to display on the mobile device screen only the information that the device is able to visualize. The entire system runs over a multi-agent platform. Considering the need to run remote agents representing the mobile devices, we chose the well-known multi-agent system (MAS) JADE [1], with the extension for mobile devices named LEAP [2].

A field service engineer (FSE) which wants to access the enterprise knowledge base, connects his mobile device to the available network infrastructure (GPRS, UMTS, Wi-Fi, Wi-Max) and contacts (through his mobile agent) a remote agent running on the server platform. This remote agent (called *interface agent*) processes the query and forwards the related tasks to the agents in charge of the databases management and the documents collection. The interface agent then collects the results of these tasks, generates an html page considering the display capabilities of the mobile device, and sends it to the FSE.

The role of D'Appolonia in E-Support is to manage the documents collection providing advanced tools able to classify them by using clustering techniques over the usual indexing of the documents corpus.

In the following sections are illustrated the E-Support project ensemble (section II), the MAS solution (section III), and the documents retrieval system (section IV), highlighting the contribution of the agents in the project success.

II. The E-Support project

A. Project context

The first task of the project was the analysis of a market study addressed to 65 European maintenance companies from Romania, Italy, Holland, Spain, and Slovakia. Most of the maintenance companies have less than 20 workers (35.5%) or 21 to 100 workers (38.7%). The survey shows that the 47.6% of the companies covers different maintenance sectors. The 23.8% operates in the chemical sector, the 22.2% in the construction field, the 15.9% in automotive, the 15.9% in the alimentary field. Few companies work in the ICT sector (4.8%). The 89.8% handles technical data in electronic format: especially by using office tools, cad, and CMMS (Computerized Maintenance Management System). No one seems to use databases.

The communication devices used by technicians are mainly cell phones (83% of companies), notebooks (55.4%), smart phones (15.4%), PDAs (10.8%) and tablet PCs (3.1%).

The interviewed companies assert that, by using the existing tools, the detected difficulties concern the trouble to find accurate (55.4%) and complete (33.8%) information, to file the info on paper (32.3%), the lack of immediate responses from the manufacturers (30.8%), to get updated information (24.6%), and the waste of money (24.6%) and time (20.0%).

Considering the emerging context, the maintenance companies which fill out the questionnaire, judge the E-Support project useful (30.6%) or quite useful (56.9%).

On the basis of these results, the E-Support tool is conceived as to limit the costs incurred by the SMEs, to easily find significant information and especially to reach the field

service engineers on the field. For these reasons, the E-Support system is designed as a service provider managed by maintenance associations and distributed to the subscribing SMEs. Figure 1 shows the high level architecture of the system. Four major challenges are recognized, in order to provide a useful tool:

- The *document retrieval system* (named FSE-Assistant) for the management of documents and the easy discovery of them.
- The *knowledge sharing and learning* (FSE-Master) for the management of databases and the training of the workers.
- *Wireless mobile client infrastructure*: it regards the network infrastructure, the security, and the compression of transmitted data.
- The *multi-agent platform* which hosts the whole service, mobile devices included.

B. *Document retrieval system*

The aim of the document retrieval system is to allow technicians on the field to find information contained in textual documents, datasheets, cad files and pictures. This large amount of electronic documents is hosted in a file server staying at the provider server farm. The tool the E-Support project wants to delivery is not a mere search engine, able to index textual documents and to systematically return a list of files containing a word: the tool is also able to order the documents into clusters [3] namely categories (and sub-categories) which contain documents belonging the same topic. These topics are automatically selected by advanced clustering algorithms. An interesting objective is the indexing of non-textual documents. In this case is useful to manually label these files with tags. The implementation of the document retrieval system is deeply discussed in section III.

C. *Knowledge sharing and learning*

The goal is to develop an intelligent open knowledge sharing and learning system able to provide a technical training to FSEs who work on site. The FSE-Master comprehends intelligent user-profiles and personalization capabilities, in a user-friendly web-based context. The learning audience benefits from a modular step-by-step approach, following several training processes: *learning*, *practicing*, *testing*, and *assessing*. According to the E-Support essence, the field service "students" can access to the "classroom", when and where they need, through their mobile devices. The e-learning tool is enriched with the possibility to share a personal knowledge with other colleagues, offering own experiences in helping them to solve a problem.

D. *Wireless mobile client infrastructure*

The main problem in reaching the workers on the field is the network infrastructure. For this reason the E-Support system must be versatile, offering different media to connect a mobile device to the central server. Considering the low costs level of the E-Support product, oriented towards the SMEs, the mobile device is able to manage different standards, choosing among those that are available on site. The E-Support tool must tolerate several bandwidths and rate profiles, self-adapting the throughput to the current context. Several wireless technologies have been evaluated considering their speed, price, compatibility with the devices, and coverage. The E-Support project took into account existing and incoming standards [4] as: Wi-Fi, Wi-Max, UMTS, GPRS, and Edge. The main considerations regard the coverage and the speed. Wi-Fi is quite widespread in offices, factories and production areas and it has an adequate bandwidth. UMTS and GPRS cover every populated area of Europe but they denote a low bandwidth. The Wi-Max technology represents the best solution, but, at now, is not commercialized and not at all diffused. Considering these points for the E-Support system, the main wireless medium is the Wi-Fi where possible, switching to GSM standards otherwise. The Wi-Max will be monitored in order to introduce it in the system as soon as possible.

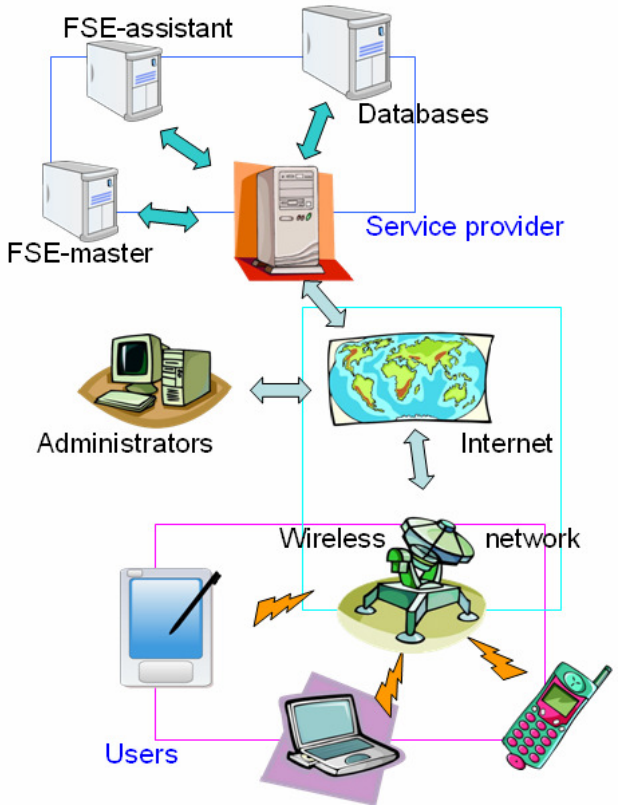

Fig. 1: the whole E-Support service.

Other issues related to the communication layer are the security of transactions and the compression of transmitted data [5]: S-HTTP and DES (Data Encryption Standard) ensure safe communications and the ZIP method to compress the exchanged files.

The connection switching and security functionalities are hidden to the end-user who can use the E-Support device without caring the communication status.

E. *Multi-agent architecture*

The motivations that encourage us to adopt a multi-agent platform concern especially:

- The dynamicity of the system, considering the mobile

devices, the different data sources, the needed intelligence and pro-activity of the requested software components.
- The scalability of the system, which has to be adapted to different scenarios, with enterprises having different sizes and requirements.
- The naturally distributed environment.

To implement the E-Support multi-agent society, the JADE platform has been chosen, due to its good reputation, stability, and mainly the possibility to distribute agents and agent containers over a network of mobile devices, by using the JADE LEAP extension. Another consideration is the fact that other E-Support components are written in Java and therefore they are easily mixable with the multi-agent platform.

The multi-agent platform is deeply investigated in the next section.

III. THE MULTI-AGENT PLATFORM

A. Introduction

In order to design a multi-agent system which accomplishes the main requirements of E-Support, some considerations are reported:

- The connection among mobile devices and the core platform is wireless.
- The connection could be slow, considering the use of different wireless technologies (GPRS, UMTS, Wi Fi, and Wi-Max).
- The connection could be affected by line losses.
- Some mobile devices could have strict restrictions to display downloaded files and information and to execute weighty programs.
- The access to centralized information is concurrent.
- The platform must be scalable and adaptable to different kind of companies, also by using pre-existent tools and databases.
- Different programmer teams will work on the system. Agent roles help the teams to integrate their modules. Ontologies modeling agent interactions are a tool that improves the integration of these service modules.

For these reasons, the architecture of the system takes into account that every hard computation is in charge of the server side, letting the mobile device free (especially if it is a PDA or any other device with limited resources). The mobile device hosts only one agent, able to connect to the remote platform, sends queries to the system, receives the results and browses them. This agent, called *front-end* agent, represents the end-user, namely the technician on the field who wants to exploit the functionalities of the E-Support platform.

The front-end agent is able to interact with *back-end agents* hosted on the server side. Back-end agents provide differentiated services to manage databases, the indexing engine, the clustering engine, and the authentication service. These agents can be cloned and the multi-agent system server can be split into different computers, in order to increase performances or to adapt to extended companies with particular requirements. The JADE cloning function and the agent containers are useful tools, in this sense.

The front-end agent does not interact directly with the back-end agents, but leans on the *interface agent*, which offers an interface of the server side services. It is a sort of mediator which collects the queries coming from the end user and reroute them to the back-end agents. Details about the interface agent and other agents are shown in the next paragraphs.

B. Front-end agent

A front-end agent is merely a browser which displays results sent by the interface agent which is talking with the agent. Every heavy computation is on the back of core agents and especially on the back of an interface agent.

When a user wants to connect to the E-Support platform through a mobile device, the corresponding front-end agent contacts the *Directory Facilitator* (DF, the yellow pages service, embedded in JADE) of the platform to discover the name of a free interface agent. The DF responds and the front-end agent opens a conversation with the suggested interface agent. Then, the user must authenticate himself, sending to the interface agent his username and password. Automatically, the front-end agent communicates the mobile device type too, in order to send, in response to the incoming queries, only the displayable information.

The user is now able to query the system. Every communication is in charge of the interface agent, which routes requests to the core agents. Every front-end agent has different behaviours, differentiated by user privileges. The intention is to consider every mobile device at disposal of each technician on the field, regardless of his rank. Every agent has a complete set of behaviours, raised only if the current user has the required rights.

Regarding the front-end agent, JADE LEAP allows programmers to split the related agent container to a front-end, hosted on the mobile device and a back-end hosted on the server, in order to move the infrastructure of the agent container on the server side, avoiding an overload of the mobile device.

C. Interface agent

The main task of an interface agent is the mediation among front-end agents and core agents. The aim of interface agents is to book core agents just the time necessary to serve out the query, without waste of time due to slow connections and line losses. The interface agent collects the queries of the front-end agent and redirects them to the core agents, then it gets the query result and releases the core agent; achieved the information, the interface agent composes a presentation using html tags, considering only those records that can be displayed by the mobile device. The information about the type of the mobile device involved in the communication can be retrieved, asking the authentication agent.

Every interface agent has more behaviours, every one specialized to interact with a core agent. Depending on the

company size and the number of field technicians, the interface agent can be configured to serve only one front-end agent at a time or to serve a strict number of front-end agents at the same time.

D. Authentication agent

It manages user's credentials and maintains his status (i.e. if he is online and his mobile device type). The information about the agent status is useful to compose a presentation page taking into account the display capabilities of the mobile device. For this reason, the interface agent contacts the authentication agent to get these data, at the moment of the composition of an html page.

E. Database agents

They manage the knowledge base of the company. The database agents interrogate the database using SQL queries.

Implementing different behaviours, it is possible to manage different types of database, considering the pre-existent tools at the company's disposal (e.g. Oracle, SQL Server…). The database contains information about customers, vendors, suppliers, parts, components and data supporting e-learning tools.

F. Text indexing agents

They index large amount of documents (*doc*, *pdf*, *txt*, *html*, etc.) reading selected folders and downloading updated versions of manuals from suggested web sites.

These agents (one or more, depending on the company size and possible specializations) write the indexing results to a centralized index, managed by a specialized agent, with the main task to coordinate the concurrent access to the centralized index. We distinguish among agents that read text documents hosted in a file server and that read web pages.

G. Clustering agent

The clustering agent processes the index created by indexing agents, in order to execute the query sent by remote users (via the interface agent). It analyzes the index and returns a list of clusters, also considering an ontology of relevant terms concerning a certain domain. The returned list of clusters and documents is raw and must be processed by the interface agent, to erase unreadable files (considering the mobile device type) and to present the result in human readable template adaptable to different kind of displays. More details about clustering agent and its services are reported in section IV.

H. Web agent

The web agent is in charge to manage interactions between the platform and the customer, who can get information about the status of the maintenance process or submit a new fix request.

The agent submits queries to the interface agent, as a normal end-user. The customer can also interact with the maintenance company through usual channels, as the telephone or email. In this case is the human operator to insert o to communicate data about a fix action.

I. Other agents

The previous agents are the most significant ones, but we can consider the introduction of other agents in order to manage integration with existent CMS (Content Management System) and other management tools. These agents denote behaviours which allow conversion among the E-Support internal knowledge representation and third-parts knowledge bases. They are easy to introduce in the MAS, if we consider them as a sort of "special" end users interacting with the interface agents and then the core agents.

J. Related works

Due to the complex nature of the whole E-Support project, it is difficult to find and describe similar works that involve every aspect of this project. Relating the document retrieval system and multi-agent technology, several proposals are described in literature. The management of electronic documents is in turn, a complex problem that involves issues as scalability, high dimensionality, content meaning, etc.

The split of the entire complex problem into circumscribed aspects is a common approach and the multi-agent system technology is a valid help to solve and manage these contexts.

A solution aimed to multimedia documents is proposed in [13] where the main goal is to efficiently accesses a scene of a video without a brute force approach (by using forward and rewind functions). As an example is reported the recording of a conference event; the system, based on specialized agents, will be able to select a scene given a query like ("give me the scene where T presents its paper). The provided solution is to define agents specialized to process single media (video, audio, text) and other agents able to locate a face in a frame, to identify the selected face, and to check if the person is speaking. An agent is able to analyse the audio in order to extract intelligible words (or, more easily, to analyse a text describing a scene provided by a human operator). The orchestration of these agents allows the system to perceive the goal.

Another solution aimed to the indexing of document in a team of users (both end-users and owners of documents) is described in [14]. An agent runs on every team member PC; it is able to maintain an updated index of data stored in the PC and to reply to the queries given by the team member. This agent contacts its counterparts running on the other members' PCs, in order to support a parallel document search. Considering the architecture of the team network, the agent running on a PC interacts with peers synchronously where possible, asynchronously (via mail) where firewalls or other systems inhibit the direct communication.

In [15] is illustrated a prototype which is quite similar with the E-Support document retrieval system. The system offers facilities to index and share information both in local repositories and in the WWW. Every user can organize the retrieved information in hierarchical categories (a sort of static cluster) with the help of a personal agent which is able to interact with other personal agents in order to share local data among end-users. Other agent roles are defined: the matchmaker agents help personal agent to find the peers that manage significant information for a particular end-user query.

Group agents are particular personal agents that manage common repositories, not directly linkable to a specific end-user. Finally the knowledge agent (that is under development) has the main feature to manage knowledge bases related to particular domains in order to improve the results of end-user queries.

Regarding the clustering techniques involving multi-agent systems, a proposed solution [9] split the clustering process, usually centralized, in a distributed environment, where clustering agents manage local repositories and learn from the results of cooperative peers.

Another proposed system involves swarm intelligence and in particular the self-organization behaviour of ants applied to large amount of documents.

K. *A demonstrative scenario*

In order to explicate the main functions of these agents, a simple interaction of a remote user is reported (figure 3).

A technician on the field must access to the company knowledge base to search information about a *hot-water heater* to repair.

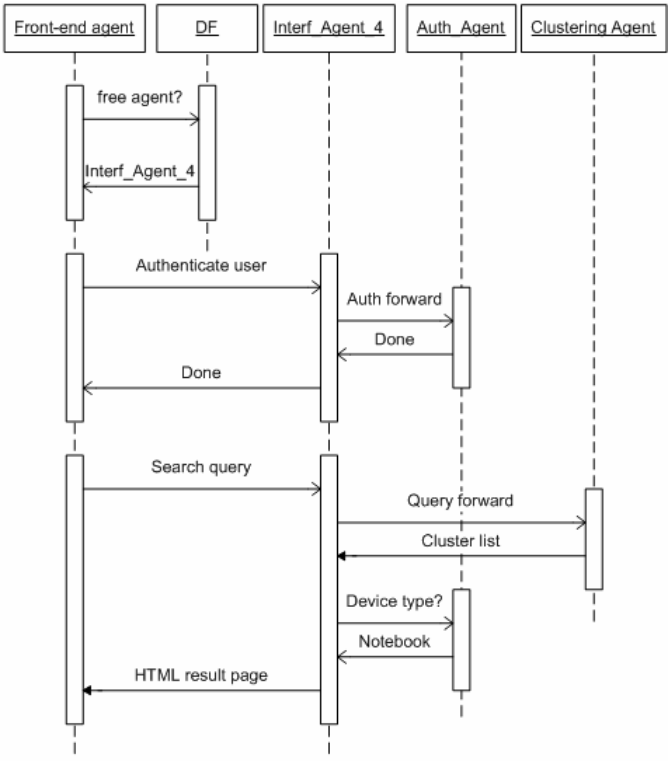

Fig. 2: an example of agent interactions.

He is on the field, namely in the boiler room, so he can connect to the Wi-Fi LAN of the office. Through Internet, the front-end agent contacts the remote server and asks the DF in order to find a free interface agent (see in figure 2 the agent interactions for a search query). Established the channel between front-end agent and interface agent, the end-user must authenticate himself. The front-end agent sends username, password and mobile device type to the interface agent which reroutes the information to the authentication agent. Once the user is logged, he can send queries to the remote system. As an example he wants to find all the documents containing the word "heater". He types the word "heater" and its front-end agent sends the query to the interface agent.

The interface agent forwards the query to the clustering agent. Ignoring for the moment the detailed functioning of the information retrieval system, the clustering agent reads the index of documents owned by a particular indexing agent and then returns a hierarchical list of categories containing all the documents belonging to knowledge base of the company, in which is contained the word "heater". The categories represent different topics regarding a heater, depending on the occurrence of most significant words in every document. If the mobile device used by the user does not support a type of document (for example *ps* files), the interface agent prunes from the clustering list all the *ps* files. Therefore, the cluster list is converted by the interface agent in an html page, ready to be sent to the front-end agent.

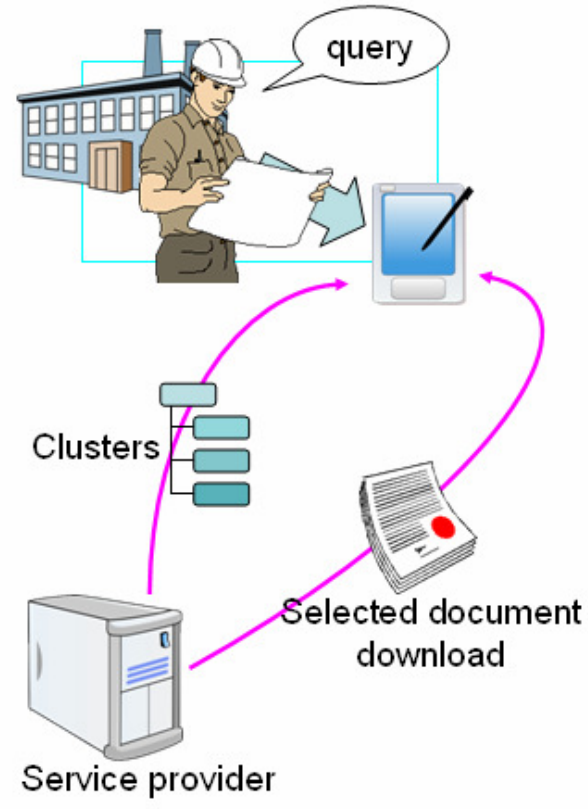

Fig. 3: the remote connection of a worker.

The technician is now able to navigate the clusters, in order to select the most relevant documents and download them.

Read the documents, for example technical manuals, the user decides that the heater must be repaired, replacing a particular component. In this case, the technician can interrogate the central database to discover the closest supplier of this component, its price, and relative instructions. As usual, the front-end agent contacts the interface agent, which reroutes the queries to the apposite database and composes the html page containing the results.

IV. DOCUMENT RETRIEVAL SYSTEM

A. Introduction

The contribution of D'Appolonia in the E-Support project concerns the *document retrieval system*. As mentioned in the previous sections, this system is able to receive the queries of the user, in order to find documents and every other useful file for the everyday work activity. The goal is to provide a smart system which allows the indexing of the entire document corpus of the company and the automatic clustering of documents on the basis of the searched word. E-support wants to ease the work of technicians and engineers on the field and this essence is also applied to the document retrieval system. The use of clustering techniques represents a first automatic partition of documents in the most relevant topics concerning a particular maintenance field. The solution is reported in the next paragraphs.

B. Clustering techniques

The document clustering is an unsupervised learning technique that furnishes a hierarchy of document which facilitates the browsing of a collection of documents. Documents belonging the same cluster have a high degree of similarity. In general, during the clustering generation process, a document is represented by a vector that contains the most significant words of the selected document. A pre processing process could be useful to remove stop words, forbidden words, etc. Some clustering techniques allows the extrapolation of the most significant words considering the entire document set, in order to focus a set of expressions with a significant discriminating power. Meaningful elements to classify a clustering algorithm are the following:

- High dimensionality: usually, in a document there are thousands or tens of thousands relevant terms. Each term represents a dimension of the document. Natural clusters are not selected in the full dimensional space, but in subspaces formed by a set of correlated dimensions. The identification of these subspaces is often difficult.
- Scalability: the algorithms must work fine with both limited and large sets of documents.
- Accuracy: documents belonging the same cluster must be similar. External evaluation methods for the accuracy measure are developed [10].
- Meaningful cluster description: every cluster must be described with a significant label which helps the user to browse the cluster hierarchy.
- Prior domain knowledge: several algorithms can be tuned with input parameters. Oftentimes, the user is unable to set up these parameters. In this case the algorithms should not sensitively decrease the performances.

Clustering algorithms can be classified into main categories:

- Hierarchical clustering methods: they can have a bottom-up approach, i.e. they generate a set of clusters and then they merge the most similar clusters; the top-down approach, on the other hand, divides every cluster in sub-clusters, until an end condition is reached.[11]
- Partitional clustering methods: they consist in the k-means methods and variants. Every document is associated to the closest centroid (considering the similarity), starting from a set of k random centroids. The algorithm selects a centroid to split and repeats the process, until the number of k cluster is reached. [11]
- Frequent item-set based methods: the idea is that many frequent items (namely terms) should be shared within a cluster and different cluster should have more or less different frequent items. The result is not a hierarchical clustering; in order to introduce this feature, the notion of item-set is created. [12]

C. Architecture

The figure 4 shows the proposed architecture of the system. The whole system is based on the collection of textual documents and other type of files (as pictures, cad files, datasheets, etc.). These documents can be tagged, in order to add useful information to the single file. This is helpful for textual documents, in order to add detailed information about the nature of the document; this option becomes necessary in case of non-textual documents. These tags are defined in specific ontologies: one for the description of document types and one containing the relevant entities of a particular maintenance sector (for example computer maintenance, electrical, heating, building, etc.). The administrator of the system or a skilled employee can create an instance of a particular concept and then attach it to the selected document.

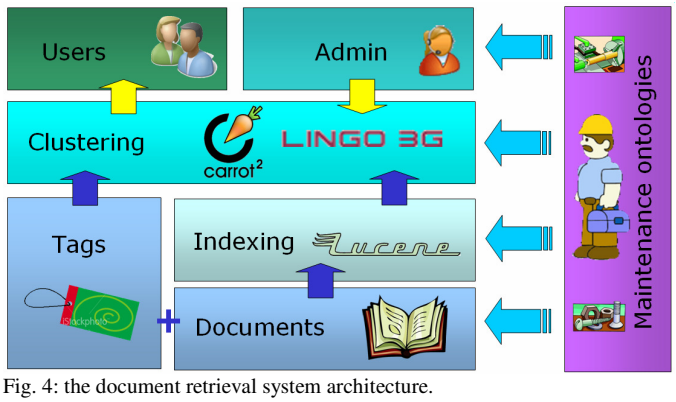

Fig. 4: the document retrieval system architecture.

The textual documents are indexed by the *Apache Lucene* indexing engine. An indexing agent continuously reads the document corpus and updates the generated index. Another similar agent supervises a group of selected web sites and the result of the web indexing is merged with the main one.

The clustering engine taps into the Lucene index and the set of tags in order to provide a more efficient clustering. For the E-Support we select the *Carrot²* clustering engine, which implements the most advanced clustering algorithms.

Authenticated users exploit the functionalities of Carrot² and administrators are able to tune the parameters of Carrot² and configure the whole system.

D. The document corpus

The work of a maintenance technician is not only based on the hint of textual documents. Maintenance companies provide precise requirements regarding the sharing of pictures of plants and parts, with attached comments containing the experiences of the colleagues. These companies consider helpful the indexing of cad files containing the technical schema of the plants, datasheet with details of the components, and maps of buildings, compounds and cities.

E. Tags and ontologies

An ontology containing the different types of documents is going to be developed. The aim of this ontology is to provide a complete catalogue of the possible documents owned by a maintenance enterprise. For example, the document ontology contains the *textualDocument* class, with properties *hasTitle*, *hasTopics*, and *hasComment*. The textual document class is specialized in different subclasses, e.g. the *book* class with properties *hasTitle*, *hasTopics*, *hasComment* (all inherited from the textual document class), *hasISBN*, *hasEditor*; the webpage class with properties *hasTitle*, *hasTopics*, *hasComment*, *hasURL* etc. Another class (disjoint from the *textualDocument* class) could be the *picture* class, with a property for the technician comments, the location, the date etc.

Regarding the maintenance ontologies, a generic maintenance ontology will be delivered and different domain specific ontologies will be built on the basic one. The survey involving European maintenance companies shows that these companies cover different maintenance sectors and often the same enterprise manages several fields. Therefore, specific ontologies are the solution in order to deeply customize the functioning of the document retrieval system, especially increasing the performances of the clustering engine.

For example, if the company does maintenance in the computer field, an ontology based on Information Technology will be built. The ontology will contain the most relevant entities of the computer discourse domain. The components of a PC will be described. As an example a class describing a chip of RAM, handles properties like *hasManufacturer*, *hasCapacity*, *hasDimension* etc.

Both the ontologies are used to catalog a document. For instance, we have a web page containing specific information about RAM chips and then the human operator can add a series of tags describing the page: a tag that instances the concept of the *web page*, one or more tags instantiating the concept RAM, and so on. The ontologies are developed by using the Protégé ontology editor [13] and the OWL language [14].

F. The indexing engine

To furnish a valid tool for the document clustering, the whole system must be based on a robust indexing engine, able to index efficiently a large amount of files, customizable, open source and possibly written in Java. The Apache Lucene project corresponds perfectly to these requirements. Lucene offers good performances in terms of RAM and CPU usage and disk space occupation. It allows the definition of customized file parsers in order to read every type of file. The complete API of Lucene allows the implementation of an agent able to manage the indexing process and the management of the resulting index. The indexing engine provides functions to merge different indexes, to build an index in incremental mode or batch mode; it allows the simultaneous searching and updating and the research through several fields as the title, the author, the content, etc.

G. The clustering engine

It is the core of the document retrieval system. The selected tool is the Carrot² clustering engine. It is a Java open source software that automatically organizes the search results into thematic categories. At now Carrot² supports 5 algorithms: Fuzzy Ants [15], HAOG-STC, Lingo [16], Rough K-Means [17], STC [18]. The clustering engine can exploit different indexing engines both online as Google, MSN, Yahoo, and offline as Lucene.

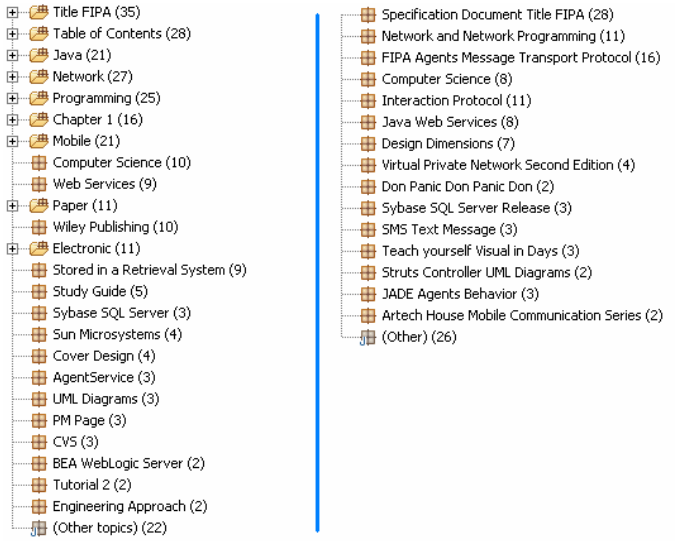

Fig. 5: the clusters generated by Lingo3G compared with the Carrot² ones.

There exists a commercial version of Carrot², named Lingo3G, which provides improved features and performances. It is able to get hierarchical clusters, to filter or boost suggested cluster names, supports the definition of synonyms and multilingual clustering. The APIs of the free version and the commercial one are quite similar, then, in the E-Support project, we intends to provide both the services and allow the companies to chose the level of accuracy they need buying the license of the commercial version or using the open source one. In parallel with the Carrot²/Lingo3G clustering we execute a sort of clustering based on the existing tags. The clustering based on tags is more accurate, because it derives from a human classification. Even if the automatic clustering is less accurate and smart, the results of the tests we did are encouraging. The results of the two clustering processes are merged, obviously highlighting the ones coming from the tag clustering.

Another interesting feature connected to the clustering is the extension of the thematic classification to the terms belonging to the maintenance ontology which are related to the current searched keyword (only if the keyword belongs to the ontology) through usual ontological relations as: *is subclass of*, *is a part of*, *same as* etc.

Fig. 6: sub-clusters for the "Java" cluster (Lingo3G).

All these features are managed by the clustering agent which runs different behaviours that implement the aforementioned methodologies of clustering.

Some testing results are shown in figures 5 and 6: the clustering engine prototype ran on an index containing about 800 indexed documents (pdf and doc file). The system returns a collection of clusters for the keyword "message". Figure 5 reports a comparison between the Lingo3G results and the Carrot² ones. Figure 6 reports a detail regarding a sub-cluster generated for the "Java" cluster. Considering these results, the Lingo3G engine returns a high number of significant clusters. The cluster labels appear more accurate and sub-clusters are on topic with the parent cluster.

H. The user functionalities

As emerges from the requirements analysis, the user (regarding the document retrieval system) is able to access the knowledge base of the company represented by the collection of documents and other files, through different ways:

- Simple search: the user enters a keyword and the system returns a set of clusters (figure 7).
- Conceptual search: the user instantiates a concept coming from the document or the maintenance ontology. The system returns another set of clusters based on the instance of the concept.
- Directories: they are a sort of static clusters: typical topics related to the specific maintenance sector. They are set *a priori*.
- Documents chronology: the list of recent documents read by the user.
- New and updated documents: a list of the new documents and a report of the updated ones.

I. The administrator features

Regarding the document retrieval system, the administrator manages the entire documents corpus. He can add, update and modify documents, and he can manage the tags linked to every file. A particular function of the administrator is the possibility to create user profiles. A user's profile describes a particular category of end users. In a maintenance enterprise the workers are specialized in a maintenance sector, therefore for each worker profile, the administrator can select a particular maintenance ontology, a particular set of documents directories, a certain list of preferred cluster labels, and a customized Lingo3G/Carrot² setup. The administrator can link every user to a profile in order to improve the performances of the document retrieval system.

Fig. 7: a demo for the simple search function.

V. CONCLUSIONS

The E-Support project is currently under development. At now, the European partners are going to validate the final architecture proposal and the implementation of the system will start as soon as possible. Regarding the document retrieval system, both the relative agents and the architecture seems to be consolidated.

Concerning the performances of the clustering engine, we are stressing the two versions: Carrot² and Lingo3G. Obviously the two tools are not a magic box and the results are less efficient than the human classification. Nevertheless the results are encouraging and fairly positive: the test tools we have developed works on thousands of technical documents and scientific papers and they have become an utility for the everyday activity. We think that through an opportune calibration, and with the help of ontologies the developed tools supporting the front-end agent, will make it a very smart agent: a useful partner for each technician on the field.

REFERENCES

[1] F. Bellifemine, G. Caire, D. Greenwood, "Developing Multi-agent Systems with JADE", Wiley, 2007.

[2] M. Berger, S. Rusitschka, D. Toropov, M. Watzke, M. Schlichte, "Porting Distributed Agent-Middleware to Small Mobile Devices", *In Proceedings of the Workshop on Ubiquitous Agents on embedded, wearable, and mobile devices*, Bologna, 2002.

[3] S. Landau, M. Leese, "Cluster Analysis", Oxford University Press US, 2001.

[4] A. Tanenbaum, "Computer Networks", Prentice Hall, 2002.

[5] D. Salomon, "Data Privacy and Security: Encryption and Information Hiding", Springer-Verlag, New York 2003.

[6] F. Dubois, B Mérialdo, "A framework for multi-agent multimedia indexing", *In Proceedings of workshop on intelligent multimedia information indexing*, August 1995 - Montreal, Canada.

[7] C. N. Linn, "A multi-agent system for cooperative document indexing and queryingin distributed networked environments", *In Proceedings of International Workshops on Parallel Processing*, 1999.

[8] J. Chen, S. Wolfe, S. Wragg, "A distributed multi-agent system for collaborative information management and sharing", *In Proceedings of the ninth international conference on Information and knowledge management***,** McLean, Virginia, United States, 2000.

[9] K. Hammouda and M. Kamel, "Collaborative Document Clustering*", In Proceedings of Conference on Data Mining (SDM06)*, pp. 453-463, Bethesda, Maryland, April 2006.

[10] C. J. van Rijsbergen, "Information Retrieval" Butterworth Ltd., second edition, London, 1979.

[11] L. Kaufman, P. J. Rousseeuw, "Finding Groups in Data: An Introduction to Cluster Analysis", New York: John Wiley & Sons, 1990.

[12] K. Wang., C. Xu, B. Liu, Clustering transactions using large items. *In Proceedings of International Conference on Information and Knowledge Management, CIKM'99*, Kansas City, Missouri, United States, 483–490, 1990.

[13] J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubezy, H. Eriksson, N. F. Noy, S. W. Tu, "The Evolution of Protégé: An Environment for Knowledge-Based Systems Development", 2002.

[14] D. L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview", Feb. 2004, [Online document], Available at HTTP: http://www.w3.org/TR/owl-features/

[15] S. Schockaert, M. de Cock, C. Cornelis, E. E. Kerre, "Efficient Clustering with Fuzzy Ants", In Proceedings of Ant Colony Optimization and Swarm Intelligence (ANTS 2004), pages 342-349, 2004.

[16] S. Osinski, D. Weiss, "A Concept-Driven Algorithm for Clustering Search Results", IEEE Intelligent Systems, 3 (vol. 20), 2005, pp. 48-54.

[17] C.L.Ngo, H.D.Nguyen, A Tolerance Rough Set Approach to Clustering Web Search Results*, in Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2004)*, Italy, September 2004.

[18] J. Stefanowski, D. Weiss, "Carrot and Language Properties in Web Search Results Clustering", *In Proceedings of the First International Atlantic Web Intelligence Conference*, Madrid, Spain, 2003, pp. 240-249.