OSM: A Multi Agent System for Modeling and Monitoring the Evolution of Oil Slicks in Open Oceans

Juan Manuel Corchado, Aitor Mata and Sara Rodríguez

Abstract. A multi agent based predicting system is presented in which the aim is to forecast the presence or not of oil slicks in a certain area of the open sea after an oil spill. In this case, the multi agent architecture incorporates a predicting system based on the CBR methodology, implemented in a series of interactive services, for modeling and monitoring the ocean water masses. The system's nucleus is formed by a series of deliberative agents acting as controllers and administrators for all the implemented services. The implemented services are accessible in a distributed way, and can be accessed even from mobile devices. The proposed system uses information such as sea salinity, sea temperature, wind, currents, pressure, number and area of the slicks. obtained from various satellites. The system has been trained using data obtained after the Prestige accident. Oil Spill Multiagent system (OSM) has been able to accurately predict the presence of oil slicks in the north west of the Galician coast, using historical data.

Keywords. Multi agent systems, Oil spill, CBR agents, PCA.

1. Introduction

The response to minimize the environmental impact when an oil spill is produced must be precise, fast and coordinated. The use of contingency response systems can facilitate the planning and tasks assignation when organizing resources, especially when multiple people are involved.
When an oil spill is produced, the response to minimize the impact must be precise, fast and coordinated. In that kind of situations, where multiple people are involved, a flexible and distributed architecture is needed in order to develop effective contingency response systems.
One of the most important characteristics is the use of intelligent agents as the

main components in employing a service oriented approach, focusing on distributing the majority of the systems' functionalities into remote and local services and applications. The architecture proposes a new and easier method of building distributed multi-agent systems, where the functionalities of the systems are not integrated into the structure of the agents, rather they are modelled as distributed services and applications which are invoked by the agents acting as controllers and coordinators.

Agents have a set of characteristics, such as autonomy, reasoning, reactivity, social abilities, pro-activity, mobility, organization, etc. which allow them to cover several needs for artificial intelligence environments[30], especially ubiquitous communication and computing and adaptable interfaces[8]. Agent and multi-agent systems have been successfully applied to several scenarios, such as education, culture, entertainment, medicine, robotics, etc. [6][25]. The characteristics of the agents make them appropriate for developing dynamic and distributed systems, as they possess the capability of adapting themselves to the users and environmental characteristics [14]. The continuous advancement in mobile computing makes it possible to obtain information about the context and also to react physically to it in more innovative ways . The agents in this architecture are based on the deliberative Belief, Desire, Intention (BDI) model [3], where the agents' internal structure and capabilities are based on mental aptitudes, using beliefs, desires and intentions . Nevertheless, modern developments need higher adaptation, learning and autonomy levels than pure BDI model [3]. This is achieved in new multi-agent architectures by modelling the agents' characteristics to provide them with mechanisms that allow solving complex problems and autonomous learning. Some of these mechanisms are Case-Based Reasoning (CBR) [1] and Case-Based Planning (CBP), where problems are solved by using solutions to similar past problems [6]. Solutions are stored into a case memory, which the mechanisms can consult in order to find better solutions for new problems. CBR and CBP mechanisms have been modelled as external services. Deliberative agents use these services to learn from past experiences and to adapt their behaviour according the context.

Predicting the behaviour of oceanic elements is a quite difficult task. In this case the prediction is related with external elements (oil slicks), and this makes the prediction even more difficult. Open ocean is a highly complex system that may be modelled by measuring different variables and structuring them together. Some of those variables are essential to predict the behaviour of oil slicks. In order to predict the future presence of oil slicks in an area, it is obviously necessary to know their previous positions. That knowledge is provided by the analysis of satellite images, obtaining the precise position of the slicks.

The solution proposed in this paper generates, for different geographical areas, a probability (between 0 and 1) of finding oil slicks after an oil spill. OSM has been constructed using historical data and checked using the data acquired during the Prestige oil spill, from November 2002 to April 2003. Most of the data used to develop OSM has been acquired from the ECCO (*Estimating the Circulation and Climate of the Ocean*) consortium [17]. Position and size of the slicks has been

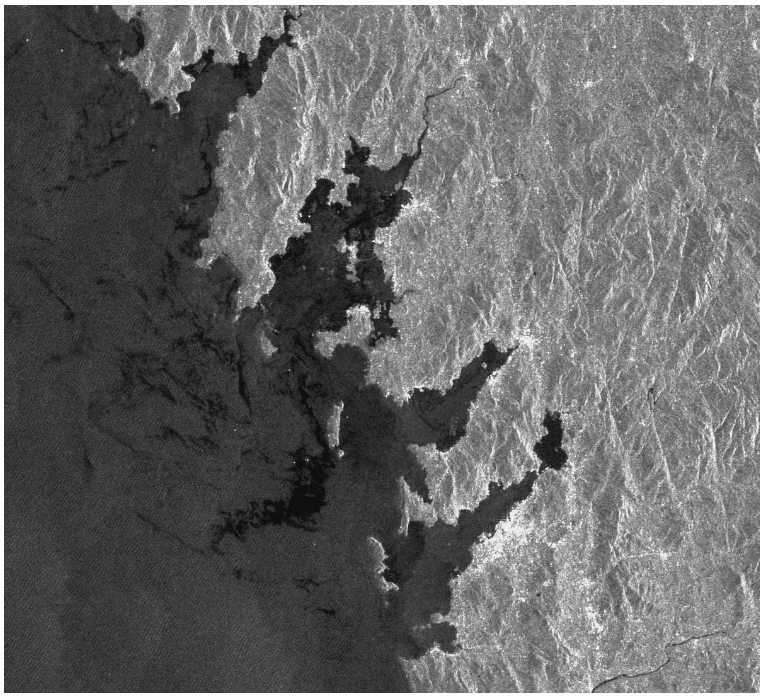

FIGURE 1. SAR image with an oil spill near the nort-west coas of Spain.

obtained by treating SAR (*Synthetic Aperture Radar*) satellite images[18].
The proposed system uses a CBR structure to learn from past situations, and to generate solutions to new problems based in the past solutions given to past problems. Past solutions are stored in the system, in the *case base*. The cases contain information about the oil slicks (size and number) as long as atmospheric data (wind, current, salinity, temperature, ocean height and pressure). OSM combines the efficiency of the CBR systems with artificial intelligence techniques in order to improve the results and to better generalize from past data. The results obtained approximate to the real process occurred in near the ninety per cent of the value of the main variables analyzed, which is a quite important approximation.
OSM allows different users to work together but without sharing the same space. The multi-agent architecture divides the system in small pieces that work separately but coordinated. The different people involved in a contingency system like the described in this paper can develop their specialized work being coordinated in the distance.
After an oil spill, it is necessary to determine if an area is going to be contaminated or not. To conclude about the presence or not of contamination in an area it is necessary to know how the slicks generated by the spill behave.

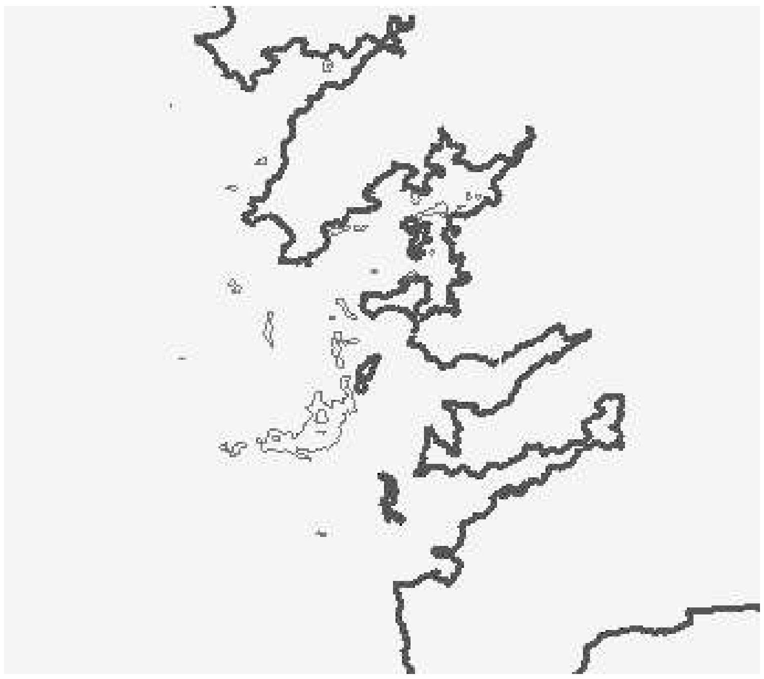

Figure 2. Interpretation of a SAR image done by the system.

First, position, shape and size of the oil slicks must be identified. One of the most precise ways to acquire that information is by using satellite images. SAR (*Synthetic Aperture Radar*) images are the most commonly used to automatically detect this kind of slicks [27]. Satellite images show certain areas where it seems to be nothing (e.g. zones with no waves) as oil slicks. Figure 1 shows a SAR image which displays a portion of the Galician west coast with black areas corresponding to oil slicks.

Figure 2 shows the interpretation of the SAR image after treating the data. SAR images make it possible to distinguish between normal sea variability and oil slicks. It is also important to make a distinction between oil slicks and look-alikes. Oil slicks are quite similar to quiet sea areas, so it is not always easy to discriminate between them. If there is not enough wind, the difference between the calmed sea and the surface of an oil slick is less evident. This can lead to mistakes when trying to differentiate between a normal situation and an oil slick. This is a crucial aspect in this problem that can be automatically managed by computational tools [24]. Once the slicks are correctly identified, it is also crucial to know the atmospheric and maritime situation that is affecting the zone at the moment that is being analyzed. Information collected from satellites is used to obtain the atmospheric data needed. That is how different variables such as temperature, sea height and salinity are measured in order to obtain a global model that can explain how slicks evolve.

There are different ways to analyze, evaluate and predict situations after an oil spill. One approach is simulation [4], where a model of a certain area is created introducing specific parameters (weather, currents and wind) and working along with a forecasting system. Using simulations it is easy to obtain a good solution for a certain area, but it is quite difficult to generalize in order to solve the same problem in related areas or new zones. It is possible to replace the oil spill by drifters to obtain a trajectory model comparing the trajectory followed by the drifters with the already known oil slicks trajectories. If the drifters follow a similar trajectory as the one that followed the slicks, then a model can be created and there will be a possibility of creating more models in different areas. Another way of predicting oil slicks trajectories is studying previous cases for obtaining a trajectory model for a certain area [28]. One step over these solutions is the use of systems that combining a major set of elements generate response models to solve the oil spill problem. A different point of view is given by complex systems which analyze large databases (environmental, ecological, geographical and engineering) using expert systems. This way, an implicit relation between problem and solution is obtained, but with no direct connection between past examples and current decisions. Nevertheless arriving at these kinds of solutions requires a great data mining effort. Once the oil spill is produced there should be contingency models for making a fast solution possible. Expert systems have also been used for solving this problem. These systems use stored information from past cases as a repository where future applications will find structured information. The final objective of all these approaches is to be decision support systems in order to enhance the response against oil spill situations. Different techniques have been used to achieve this objective, from fuzzy logic to negotiation with multi-agent systems. One of these techniques is Case-Based Reasoning which is described in the next section.

In this paper, the oil spill problem is first presented, showing its difficulties and the possibilities of finding solutions to the problem. Then, the multi-agent architecture is described. Afterwards, OSM is explained, and last, the results are shown and also the future developments that can be achieved with the system.

2. A Multi-Agent communication architecture for integrating distributed services

A multi-agent architecture has been developed to integrate the predicting services. Because the architecture acts as an interpreter, the users can run applications and services programmed in virtually any language, but have to follow a communication protocol that all applications and services must incorporate. Another important functionality is that, thanks to the agents' capabilities, the systems developed can make use of reasoning mechanisms or learning techniques to handle services and applications according to context characteristics, which can change dynamically

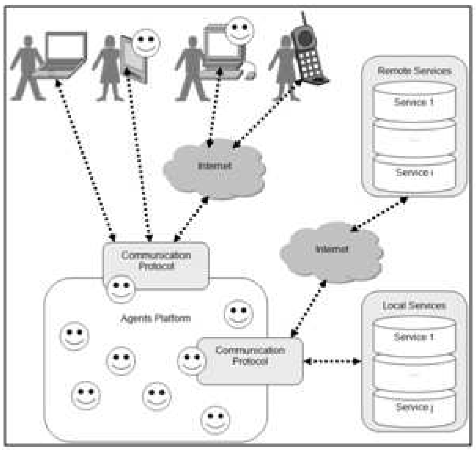

FIGURE 3. Framework basic schema.

over time. Agents, applications and services can communicate in a distributed way, even from mobile devices. This makes it possible to use resources no matter its location. It also allows the starting or stopping of agents, applications, services or devices separately, without affecting the rest of resources, so the system has an elevated adaptability and capacity for error recovery.

Users can access the system through distributed applications, which run on different types of devices and interfaces (e.g. computers, cell phones, PDA). Figure 3 shows the basic schema of the framework where all requests and responses are handled by the agents in the platform. The agents analyze all requests and invoke the specified services either locally or remotely. Services process the requests and execute the specified tasks. Then, services send back a response with the result of the specific task.

The presented framework is a modular multi-agent architecture, where services and applications are managed and controlled by deliberative BDI (Belief, Desire, Intention) agents [3][15][20][29]. Deliberative BDI agents are able to cooperate, propose solutions on very dynamic environments, and face real problems, even when they have a limited description of the problem and few resources available. These agents depend on beliefs, desires, intentions and plan representations to solve problems [2][11][22]. Deliberative BDI agents are the core of the multiagent communication system. There are different kinds of agents in the architecture, each one with specific roles, capabilities and characteristics. This fact facilitates the flexibility of the architecture in incorporating new agents. However, there are pre-defined agents which provide the basic functionalities of the architecture:

- **CommApp Agent**. This agent is responsible for all communications between applications and the platform. It manages the incoming requests from the applications to be processed by services. It also manages responses from services (via the platform) to applications. CommApp Agent is always on l̈istening mode: Applications send XML messages to the agent requesting a service, then the agent creates a new thread to start communication by using sockets. The agent sends all requests to the Manager Agent which processes the request. The socket remains open until a response to the specific request is sent back to the application using another XML message. All messages are sent to Security Agent for their structure and syntax to be analyzed.
- **CommServ Agent**. It is responsible for all communications between services and the platform. The functionalities are similar to CommApp Agent but backwards. This agent is always on l̈istening modeẅaiting for responses of services. Manager Agent signals to CommServ Agent which service must be invoked. Then, CommServ Agent creates a new thread with its respective socket and sends an XML message to the service. The socket remains open until the service sends back a response. All messages are sent to Security Agent for their structure and syntax to be analyzed. This agent also periodically checks the status of all services to know if they are idle, busy, or crashed.
- **Directory Agent**. It manages the list of services that can be used by the system. For security reasons [26], the list of services is static and can only be modified manually; however, services can be added, erased or modified dynamically. The list contains the information of all trusted available services. The name and description of the service, parameters required, and the IP address of the computer where the service is running are some of the information stored in the list of services. However, there is dynamic information that is constantly being modified: the service performance (average time to respond to requests), the number of executions, and the quality of the service. This last data is very important, as it assigns a value between 0 and 1 to all services. All new services have a quality of service (QoS) value set to 1. This value decreases when the service fails (e.g. service crashes, no service found, etc.) or has a subpar performance compared to similar past executions. QoS is increased each time the service efficiently processes the tasks assigned. Information management is especially important because the data processed is very sensitive and personal. Thus, security must be a major concern. For this reason the multiagent architecture does not implement a service discovery mechanism, requiring systems to employ only the specified services from a trusted list of services. However, agents can select the most appropriate service (or group of services) to accomplish a specific a task.
- **Supervisor Agent**. This agent supervises the correct functioning of the other agents in the system. Supervisor Agent periodically verifies the status of all agents registered in the architecture by sending ping messages. If there is no

response, the Supervisor agent kills the agent and creates another instance of that agent.

- **Security Agent**. This agent analyzes the structure and syntax of all incoming and outgoing XML messages. If a message is not correct, the Security Agent informs the corresponding agent (CommApp or CommServ) that the message cannot be delivered. This agent also directs the problem to the Directory Agent, which modifies the QoS of the service where the message was sent.
- **Manager Agent**. Decides which agent must be called by taking into account the QoS and users preferences. Users can explicitly invoke a service, or can let the Manager Agent decide which service is best to accomplish the requested task. If there are several services that can resolve the task requested by an application, the agent selects the optimal choice. An optimal choice has higher QoS and better performance. Manager Agent has a routing list to manage messages from all applications and services. This agent also checks if services are working properly. It requests the CommServ Agent to send ping messages to each service on a regular basis. If a service does not respond, CommServ informs Manager Agent, which tries to find an alternate service, and informs the Directory Agent to modify the respective QoS.
- **Interface Agent**. This kind of agent was designed to be embedded in users' applications. Interface agents communicate directly with the agents in the architecture so there is no need to employ the communication protocol, rather the FIPA ACL specification. The requests are sent directly to the Security Agent, which analyzes the requests and sends them to the Manager Agent. The rest of the process follows the same guidelines for calling any service. These agents must be simple enough to allow them to be executed on mobile devices, such as cell phones or PDAs. All high demand processes must be delegated to services.

In the next section, the contingency response system to face oil slick situations is presented, explaining how the multi-agent architecture is integrated with a CBR system in order to obtain a flexible and distributed structure.

3. OSM: a hybrid multiagent system for contingency response in oil spill situations

CBR has already been used to solve maritime problems [7] in which different oceanic variables were involved. In this case, the data collected from different observations from satellites is processed and structured as cases. The cases are the key to obtain solutions to future problems through a CBR system.

Figure 4 shows the basic structure of the OSM system, where the interfaces agents are connected to the services through the multiagent architecture. The interface agents represent the different roles the users can perform to interact with the system. The services are the different phases of the CBR cycle, that

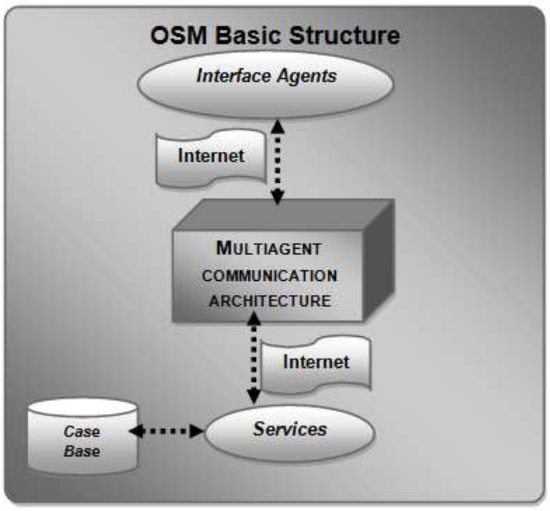

FIGURE 4. OSM basic structure.

are requested by the users. One user may only need to introduce information in the system, while expert users can be requested from the system to confirm the predictions generated.

The functionalities of the system can be accessed using different interfaces for PCs and PDAs (Personal Digital Assistant) where users can interact with the system by introducing data, requesting a prediction or revising a solution generated by the system. Figure 5 shows the main graphical user interface of OSM. The interface shows a set of parameters, the oceanic area visualization with oil slicks and a squared area to be analyzed.

Oil slicks are mainly detected using SAR images. Those images are processed and transformed to be used by the system. Oceanic, meteorological and oil spill related data is stored in the system in order to generate future predictions. The data used to train the system has been obtained after the Prestige accident, between November 2002 and April 2003, in a specific geographical area at the west of the Galician coast (longitude between 14 and 6 degrees west and latitude between 42 and 46 degrees north). Table 1 shows the basic structure of a case. The variables can be geographical (longitude and latitude), temporal (date of the case), atmospheric (wind, current, sea height, bottom pressure, salinity and temperature) and variables directly related with the problem (number and area of the slicks).

All information is stored in the case base and OSM is ready to predict future situations. A problem situation must be introduced in the system for generating a prediction. Then, the most similar cases to the current situation are retrieved from the case base. Once a collection of cases are chosen from the case base, they must be used for generating a new solution to the current problem. Growing Radial

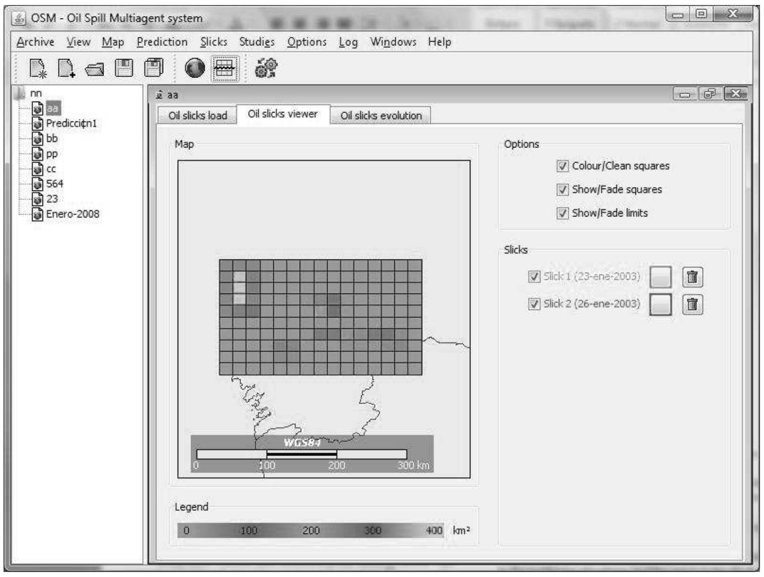

Figure 5. Graphical user interface of OSM.

Variable	Definition/Unit
Longitude	Geographical longitude/Degree
Latitude	Geographical latitude/Degree
Date	Day, month and year of the analysis/dd/mm/yyyy
Sea Height	Height of the waves in open sea/m
Bottom pressure	Atmospheric pressure in the open sea/Newton/m2
Salinity	Sea salinity/ppt(parts per thousand)
Temperature	Celsius temperature in the area/C
Area of the slicks	Surface covered by the slicks present in the area/Km2
Meridional Wind	Meridional direction of the wind/ m/s
Zonal Wind	Zonal direction of the wind/ m/s
Wind Strenght	Wind Strenght/ m/s
Meridional Current	Meridional direction of the ocean current/ m/s
Zonal Current	Zonal direction of the ocean current/ m/s
Current Strenght	Ocean current strength/ m/s

Table 1. Variables that define a case

Basis Functions Networks [16] are used in OSM for combining the chosen cases in order to obtain the new solution.

OSM determines the probability of finding oil slicks in a certain area. OSM divides the area to be analyzed in squares of approximately half a degree side for

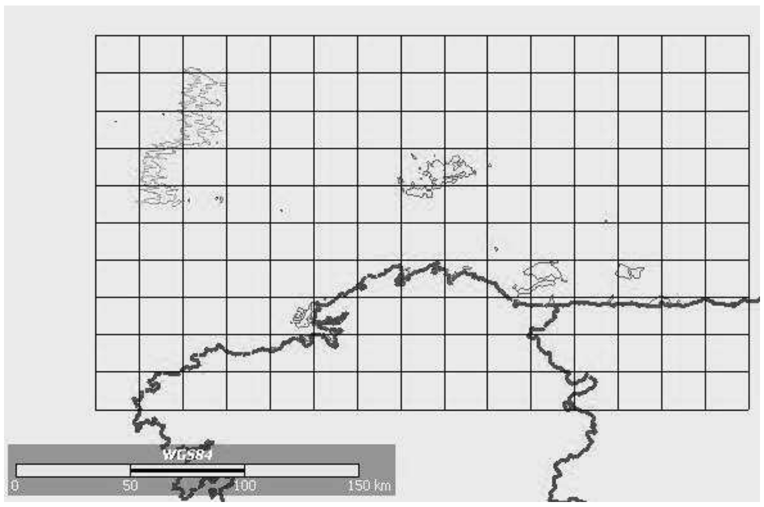

Figure 6. Division of the analyzed area into squares with slicks inside.

generating a new prediction. Then, the system determines the amount of slicks in each square. The squares are colored with different gradation depending on the quantity of oil slicks calculated.

Within the case base there are a temporal relationship between a case and its future situation. A square, with all the values of the different variables can be related with the same square but in the next temporal situation. That relationship will provide the internal mechanism used to generalize and to train the GRBF network, which will generate the prediction.

Figure 6 shows the interpretation of a series of slicks. The squared areas are those that will be analyzed by the system. First, the slicks corresponding to different days are colored in different colors. Then, in figure 7 can be seen how the squared zones are colored in different intensity depending on the amount of slicks appearing on each square (down). The bigger amount of slicks, the darker the squared is colored.

The data is stored into the case base once structured. Every case has its temporal situation stored and relates every case with the next situation in the same position. The temporal relationship creates the union between the problem and the solution. The problem is the past case, and the solution is the future case. The relationship established between a situation and its corresponding future provides the necessary data for generalizing and generating an appropriate prediction for an introduced problem.

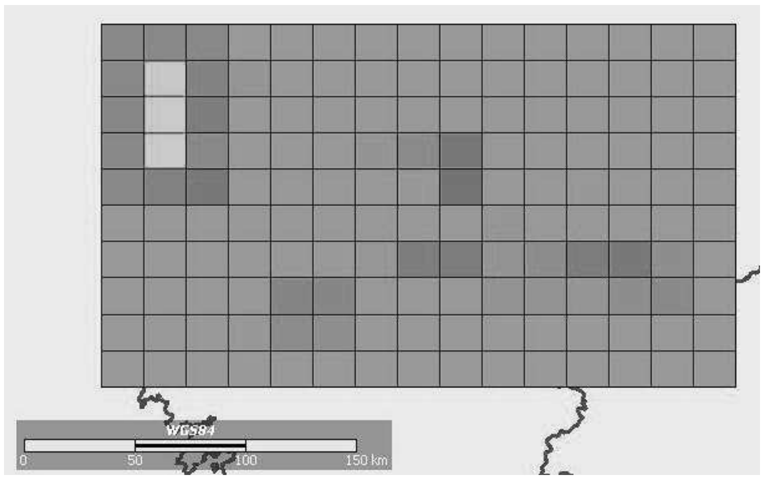

Figure 7. Interpretation of the amount of slicks in an area.

3.1. OSM architecture

OSM employs a multi-agent architecture based on SOA for distributing resources and optimizing its performance. Most of the system functionalities have been modeled as applications and services managed by deliberative BDI (Belief, Desire, Intention) agents [3][15]. Deliberative BDI agents are able to cooperate, propose solutions on very dynamic environments, and face real problems, even when they have a limited description of the problem and few resources available. These agents depend on beliefs, desires, intentions and plan representations to solve problems [11].

There are four basic blocks in OSM: Applications, Services, Agents Platform and Communication Protocol. These blocks provide all the system functionalities:

- **Applications**. These represent all the programs that users can use to exploit the system functionalities. Applications are dynamic, reacting differently according to the particular situations and the services invoked. They can be executed locally or remotely, even on mobile devices with limited processing capabilities, because computing tasks are largely delegated to the agents and services.
- **Services**. These represent the activities that the architecture offers. They are the bulk of the functionalities of the system at the processing, delivery and information acquisition levels. Services are designed to be invoked locally or remotely. Services can be organized as local services, web services, GRID services, or even as individual stand alone services. Services can make use of other services to provide the functionalities that users require. OSM has a flexible and scalable directory of services, so they can be invoked, modified, added, or eliminated dynamically and on demand. It is absolutely necessary

that all services follow a communication protocol to interact with the rest of the components.

- **Agents Platform**. This is the core of the system, integrating a set of agents, each one with special characteristics and behavior. An important feature in this architecture is that the agents act as controllers and administrators for all applications and services, managing the adequate functioning of the system, from services, applications, communication and performance to reasoning and decision-making. In OSM, services are managed and coordinated by deliberative BDI agents. The agents modify their behavior according to the users' preferences, the knowledge acquired from previous interactions, as well as the choices available to respond to a given situation.
- **Communication Protocol**. This allows applications and services to communicate directly with the Agents Platform. The protocol is completely open and independent of any programming language. This protocol is based on SOAP specification to capture all messages between the platform and the services and applications [5]. Services and applications communicate with the *Agents Platform* via SOAP messages. A response is sent back to the specific service or application that made the request. All external communications follow the same protocol, while the communication among agents in the platform follows the FIPA Agent Communication Language (ACL) specification. This is especially useful when applications run on limited processing capable devices (e.g. cell phones or PDAs). Applications can make use of agents platforms to communicate directly (using FIPA ACL specification) with the agents in OSM, so while the communication protocol is not needed in all instances, it is absolutely required for all services.

Agents, applications and services in OSM can communicate in a distributed way, even from mobile devices. This makes it possible to use resources no matter its location. It also allows the starting or stopping of agents, applications, services or devices separately, without affecting the rest of resources, so the system has an elevated adaptability and capacity for error recovery. Users can access to OSM functionalities through distributed applications which run on different types of devices and interfaces (e.g. computers, PDA). Figure 8 shows the structure of OSM. As can be seen, most of the functionalities, including the CBR system, have been modeled as services and applications. Thus, each service can be performed on demand and can also be replicated to respond multiple requests.

Interface Agents are a special kind of agents in OSM designed to be embedded in users' applications. These agents are simple enough to allow them to be executed on mobile devices, such as cell phones or PDAs because all high demand processes are delegated to services. OSM defines three different *Interface Agents*:

1. *Input Agent*. It is the agent that sends the information introduced by the users to OSM. Once the data have reached the system, it is structured into the case base. This interface agent is used by users that have visualized an oil

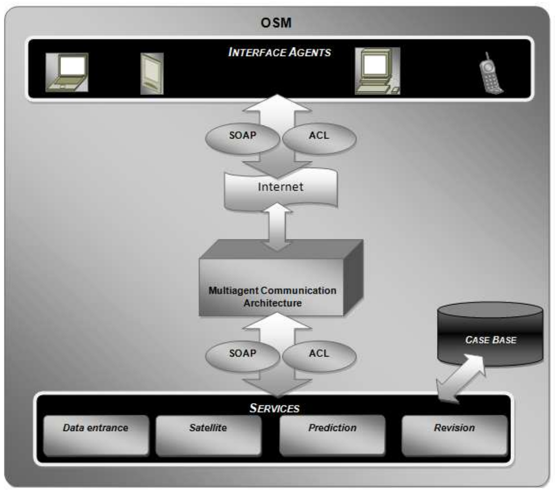

Figure 8. OSM extended structure.

slick, in order to introduce the data related with that slick into the system. the Input Agent interface is used by the users to introduce the data. The main parameters to identify the slick are its position, in terms of longitude and latitude, the surface covered by the slick, and the distance of that slick to the coast. Once the basic information about the slick has been sent to the system, OSM recovers satellite information about the ocean and the meteorological conditions in the area to create a case from the slick and geographical information.

2. *Prediction Agent*. When a user wants to request a prediction from OSM, this is the agent used to do so. In the interface of the agent, the user can define the area to be analyzed, the size of the squares to be transformed into cases and, if there are previous information stored in the system, the existing slicks to be considered to generate the prediction.
3. *Revision Agent*. When a prediction is generated by OSM, the system can automatically verify the correction of the proposed solution. But, if there are revision experts available, it also requests an expert for a revision. The users re ceive the proposed solution and enough data to validate the solution for the current problem.

OSM also defines three different services which perform all tasks that the users may demand from the system. All requests and responses are handled by the agents. The requests are analyzed and the specified services are invoked either locally or remotely. Services process the requests and execute the specified tasks. Then, services send back a response with the result of the specific task. In this way, the agents act as interpreters between applications and services in OSM.

Next, the main services defined in OSM are explained, following the main phases of the CBR cycle.

3.2. Prediction System

OSM is a Contingency Response system for Oil Spills conceived as a multi-agent system which core working structure follows the Case-Based Reasoning methodology. The different services implemented by the OSM system cover the four main phases of the OSM cycle, and also the pre-processing stage, covered by the *Data Input Service* covers. The retrieval and reuse phases are implemented in the *Prediction Generation Service* that generates a prediction after a problem description is introduced in the system by an user. The *Revision Service* covers the *revision* phase, and may require the confirmation of an expert, to validate the correction of the solution proposed. The final stage of the CBR cycle, the *retention* phase is also implemented in the *Data Input Service*, where

Data Input Service

When data about an oil slick is introduced in the system OSM must complete the information about the area including atmospheric and oceanic information: temperature, salinity, bottom pressure, sea height. All that complementary data is collected from satellite services that offer on-line and in real time that precise information. With all that information the case is created and introduced in the case base.

Historical data collected from November 2002 to April 2003 has been used to create the case base of OSM. As explained before, cases are formed by a series of variables. Principal Components Analysis (PCA) [9] can reduce the number of those variables and then, the system stores the value of the principal components, which are related with the original variables that define a case. PCA has been previously used to analyze oceanographic data and it has proved to be a consistent technique when trying to reduce the number of variables [21]. OSM uses Fast Iterative Kernel PCA (FIKPCA) which is an evolution of PCA [12]. This technique reduces the number of variables in a set by eliminating those that are linearly dependent, and it is quite faster than the traditional PCA.

To improve the convergence of the Kernel Hebbian Algorithm used by Kernel PCA, FIK-PCA set η_t proportional to the reciprocal of the estimated eigenvalues. Let $\lambda_t \in \Re^r_+$ denote the vector of eigenvalues associated with the current estimate of the first r eigenvectors. The new KHA algorithm sets de i^th component of η_t to the files.

$$[\eta_t]_i = \frac{1}{[\lambda_t]_i} \frac{\tau}{t+\tau} \eta_0 \tag{3.1}$$

The final variables are, obviously, linearly independent and are formed by combination of the previous variables. The values of the original variables can be recovered by doing the inverse calculation to the one produced to obtain the new variables. The variables that are less used in the final stored variables are those whose values suffer less changes during the periods of time analysed (salinity, temperature and pressure do not change from one day to another, then, they can be ignored considering that the final result does not depend on them). Once applied the FIKPCA, the number of variables is reduced to three, having the following distribution:

```
    Variable_1: -0,560 * long - 0,923*lat + 0,991*s_height +
0,919*b_pressure + 0,992*salinity + 0,990*temp -
0,125*area_of_slicks + 0,80*mer_wind + 0,79*zonal_wind +
0,123*w_strenght + 0,980*mer_current + 0,980*zonal_current
+ 0,980*c_strength

Variable_2: 0,292*long - 0,081*lat - 0,010*s_height -
0,099*b_pressure - 0,011*salinity - 0,013*temp -
0,021*area_of_slicks + 0,993*merl_wind + 0,993*zonal_wind
+ 0,989*w_strenght - 0,024*mer_current - 0,024*zonal_current
- 0,024*c_strength

Variable_3: 0*long - 0,072*lat + 0,009*s_height +
0,009*b_pressure + 0,009*salinity + 0,009*temp +
0,992*area_of_slicks + 0,006*mer_wind + 0,005*zonal_wind
+ 0,005*w_strenght - 0,007*mer_current - 0,007*zonal_current
- 0,007*c_strength
```

After applying FIKPCA, the historical data is stored in the case base, and is used to solve future problems using the rest of the CBR cycle. Storing the principal components instead of the original variables implies reducing the amount of memory necessary to store the information in about a sixty per cent which is more important as the case base grows. The reduction of the number of variables considered also implies a faster recovery from the case base.

When introducing the data into the case base, Growing Cell Structures (GCS) [10] are used. GCS can create a model from a situation organizing the different cases by their similarity. If a 2D representation is chosen to explain this technique, the most similar cells (i.e. cases) are near one of the other. If there is a relationship between the cells, they are grouped together, and this grouping characteristic helps the CBR system to recover the similar cases in the next phase. When a new cell is introduced in the structure, the closest cells move towards the new one, changing the overall structure of the system. The weights of the winning cell ω_c, and its neighbours ω_n, are changed. The terms ϵ_c and ϵ_n represent the learning rates for the winner and its neighbours, respectively. x represents the value of the input vector.

$$\omega_c\,(t+1) = \omega_c\,(t) + \epsilon_c\,(x - \omega_t) \tag{3.2}$$

$$\omega_n\,(t+1) = \omega_n\,(t) + \epsilon_n\,(x - \omega_n) \tag{3.3}$$

The pseudocode of the GCS insertion process is shown below:

```
1. The most similar cell to the new one is found.
2. The new cell is introduced in the middle of the connection
   between the most similar cell and the least similar to the new one.
3. Direct neighbours of the closest cell change their values by
   approximating to the new cell and specified percentage of the
   distance between them and the new cell.
```

Once the case base has stored the historical data, and the GCS has learned from the original distribution of the variables, the system is ready to receive a new problem.

When a new problem comes to the system, GCS are used once again. The stored GCS behaves as if the new problem would be stored in the structure and finds the most similar cells (cases in the CBR system) to the problem introduced in the system. In this case, the GCS does not change its structure because it has being used to obtain the most similar cases to the introduced problem. Only in the retain phase the GCS changes again, introducing the proposed solution if it is correct.

Prediction Generation Service

When a prediction is requested by a user, the system starts recovering from the case base the most similar cases to the problem proposed. Then, it creates a prediction using artificial neural networks.

The similarity between the new problem and the cases is determined by the GCS. Every element in the GCS has a series of values (every value corresponds to one of the principal components created after de PCA analysis). The distance between elements is a multi-dimensional distance where all the elements are considered to establish the distance between cells. After obtaining the most similar cases from the case base, the cases are used in the next phase. The most similar cases stored in the case base will be used to obtain an accurate prediction according to the previous solutions related with the selected cases.

Once the most similar cases are recovered from the case base, they are used to generate the solution. The prediction of the future probability of finding oil slicks in an area is generated using an artificial neural network, with a hybrid learning system. An adaptation of Radial Basis Functions Networks are used to obtain that prediction [13]. The chosen cases are used to train the artificial neural network. Radial Basis Function networks have been chosen because of the reduction of the training time comparing with other artificial neural network systems, such as Multilayer Perceptrons. In this case, the network is trained in every analysis using only the cases selected from the case base.

Growing RBF networks [23] are used to obtain the predicted future values corresponding to the proposed problem. This adaptation of the RBF networks allows the system to grow during training gradually increasing the number of elements (prototypes) which play the role of the centers of the radial basis functions. The creation of the Growing RBF must be made automatically which implies an adaptation of the original GRBF system. The error for every pattern is defined by:

$$e_i = l/p * \sum_{k=1}^{p} ||t_{ik} - y_{ik}|| \tag{3.4}$$

Where t_{ik} is the desired value of the k_{th} output unit of the i_{th} training pattern, y_{ik} the actual values ot the k^{th} output unit of the i_{th} training pattern.

The pseudo code of the Growing RBF process id described next:

```
1. Calculate the error, $e_i$ (4) for every new possible prototype.
        a. If the new candidate does not belong to the chosen
   ones and the error calculated is less than a threshold error,
   then the new candidate is added to the set of accepted
   prototypes.
        b. If the new candidate belongs to the accepted ones
   and the error is less than the threshold error, then modify the
   weights of the neurons in order to adapt them to the new
   situation.
2. Select the best prototypes from the candidates.
        a. If there are valid candidates, create a new cell
   centered on it.
        b. Else, increase the iteration factor. If the iteration
   factor comes to the 10% of the training population, freeze the
   process.
3. Calculate global error and update the weights.
        a. If the results are satisfactory, end the process. If
   not, go back to step 1.
```

Once the GRBF network is created, it is used to generate the solution to the proposed problem. The solution will be the output of the network using as input data the selected cases from the case base.

Revision Service

After generating a prediction, the system needs to validate its correction. The system can also query an expert user to confirm the automatic revision previously done. The prediction is shown to the users in a similar way the slicks are interpreted by OSM. A set of squared colored areas appear. The intensity of the color corresponds with the possibility of finding oil slicks in that area. The areas colored with a higher intensity are those with the highest probability of finding oil slicks in them. In this visual approximation, the user can check if the solution is adequate. The system also provides an automatic method of revision that must be also checked by an expert user which confirms the automatic revision.

Explanations are a recent revision methodology used to check the correction of the solutions proposed by CBR systems [19]. Explanations are a kind of justification of the solution generated by the system. To obtain a justification to the given solution, the cases selected from the case base are used again. As explained before, we can establish a relationship between a case and its future situation. If we consider the two situations defined by a case and the future situation of that case as two vectors, we can define a distance between them, calculating the evolution of the situation in the considered conditions. That distance is calculated for all the cases retrieved from the case base as similar to the problem to be solved. If the distance between the proposed problem and the solution given is not greater than the average distances obtained from the selected cases, then the solution is a good one, according to the structure of the case base. Next, the explanation pseudo code is showed:

```
1. For every selected case in the retrieval phase, the distance
   between the case and its solution is calculated.
2. The distance between the proposed problem and the proposed
   solution is also calculated.
3. If the difference between the distance of the proposed solution
   and those of the selected cases is below a certain threshold
   value, then the solution is considered as a valid one.
4. If not, the user is informed and the process goes back to the
   retrieval phase, where new cases are selected from the case base.
5. If, after a series of iterations the system does not produce a
   good enough solution, then the user is asked to consider the
   acceptance of the best of the generated solutions.
```

The distances are calculated considering the sign of the values, not using its absolute value. This decision is easily justified by the fact that is not the same to move to the north than to the south, even if the distance between two points is the same. If the prediction is considered as correct it will be stored in the case base, and it can then be used in next predictions to obtain new solutions.

The distances are calculated considering the sign of the values without using its absolute value. This decision is justified by the fact that is not the same to move to the north than to the south, even if the distance between two points is the same. If the prediction is considered correct, it will be stored in the case base and can then be used in next predictions for obtaining new solutions.

If the proposed prediction is accepted, it is considered as a good solution to the problem and can be stored in the case base in order to solve new problems. It will have the same category as the historical data previously stored in the system.

When inserting a new case in the case base, Fast Iterative Kernel PCA is used for reducing the number of variables used and adapting the data generated by the system. The adaptation is done by changing the original variables into the principal components previously chosen by the system. The internal structure of the case base also changes when a new case is introduced. The GCS system related with the case base structure controls its growth. The GCS system grows and improves its capability of generating good results as new knowledge is introduced in the system.

4. Results

OSM uses different artificial intelligence techniques to cover and solve all the phases of the CBR cycle. Fast Iterative Kernel Principal Component Analysis is used to reduce the number of variables stored in the system, getting about a 60% of reduction in the size of the case base. This adaptation of the PCA also implies a faster recovery of cases from the case base (more than 7% faster than storing the original variables).

To obtain a prediction using the cases recovered from the case base, Growing Radial Basis Function Networks has been used. This evolution of the RBF networks implies a better adaptation to the structure of the case base, which is organized using Growing Cell Structures. The results using Growing RBF networks instead of simple RBF networks are about a 4% more accurate, which is a good improvement.

Evaluations show that the system can predict in the conditions already known, showing better results than previously used techniques. The use of a combination of techniques integrated in the CBR structure makes it possible to obtain better result than using the CBR alone (17% better), and also better than using the techniques isolated(neural networks), without the integration feature produced by the CBR (11% better). A resume of all these improvements can be seen in figure 9.

The predicted situation was contrasted with the actual future situation. The future situation was known, as long as historical data was used to develop the system and also to test the correction of it. The proposed solution was, in most of

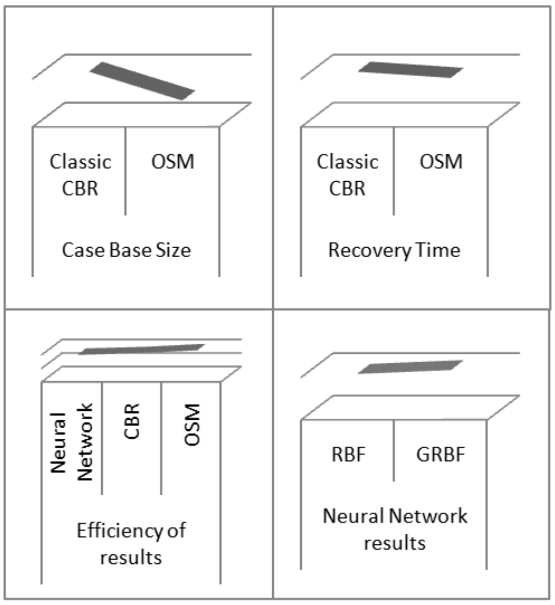

FIGURE 9. Resume of the improvement of the results obtained with OSM.

Number of cases	RBF	CBR	RBF + CBR	OSCBR
100	45%	39%	42%	43%
500	48%	43%	46%	46%
1000	51%	47%	58%	64%
2000	56%	55%	65%	72%
3000	59%	58%	68%	81%
4000	60%	63%	69%	84%
5000	63%	64%	72%	87%

TABLE 2. Percentage of good predictions obtained with different techniques.

the variables, close to 90% of accuracy. For every problem defined by an area and its variables, the system offers 9 solutions (i.e. the same area with its proposed variables and the eight closest neighbors). This way of prediction is used in order to clearly observe the direction of the slicks which can be useful in order to determine the coastal areas that will be affected by the slicks generated after an oil spill.

Table 2 shows a summary of the results obtained after comparing different techniques with the results obtained using OSM. The table shows the evolution of the results along with the increase of the number of cases stored in the case base. All the techniques analyzed improve its results while increasing the number

.	RBF	CBR	RBF + CBR	OSCBR
RBF				
CBR	*			
RBF + CBR	=	=		
OSCBR	*	*	*	

Table 3. Multiple comparison procedure among different techniques.

of cases stored. Having more cases in the case base, makes easier to find similar cases to the proposed problem and then, the solution can be more accurate. The R̈BFc̈olumn represents a simple Radial Basis Function Network that is trained with all the data available. The network gives an output that is considered a solution to the problem. The C̈BRc̈olumn represents a pure CBR system, with no other techniques included; the cases are stored in the case base and recovered considering the Euclidean distance. The most similar cases are selected and after applying a weighted mean depending on the similarity of the selected cases with the inserted problem, a solution s proposed. The R̈BF + CBRc̈olumn corresponds to the possibility of using a RBF system combined with CBR. The recovery from the CBR is done by the Manhattan distance and the RBF network works in the reuse phase, adapting the selected cases to obtain the new solution. The results of the R̈BF+CBRc̈olumn are, normally, better than those of the C̈BR;̈ mainly because of the elimination of useless data to generate the solution. Finally, the ÖSMc̈olumn shows the results obtained by the proposed system, obtaining better results that the three previous analyzed solutions.

Table 3 shows a multiple comparison procedure (Mann-Whitney test) used to determine which models are significantly different from the others. The asterisk indicates that these pairs show statistically significant differences at the 99.0% confidence level. OSM presents statistically significant differences with the rest of the models. The proposed solution does not generate a trajectory, but a series of probabilities in different areas, what is far more similar to the real behavior of the oil slicks.

Several tests have been done to compare the overall performance of OSM. The tests consisted of a set of requests delivered to the *Prediction Generation Service* (PGS) which in turn had to generate solutions for each problem. There were 50 different data sets, each one with 10 different parameters. The data sets were introduced into the PGS through a remote PC running multiple instances of the *Prediction Agent*. The data sets were divided in five test groups with 1, 5, 10, 20 and 50 data sets respectively. There was one *Prediction Agent* for each test group. 30 runs for each test group were performed. Several data have been obtained from these tests, notably the average time to accomplish the solutions, the number of crashed agents, and the number of crashed services. First, all tests were performed

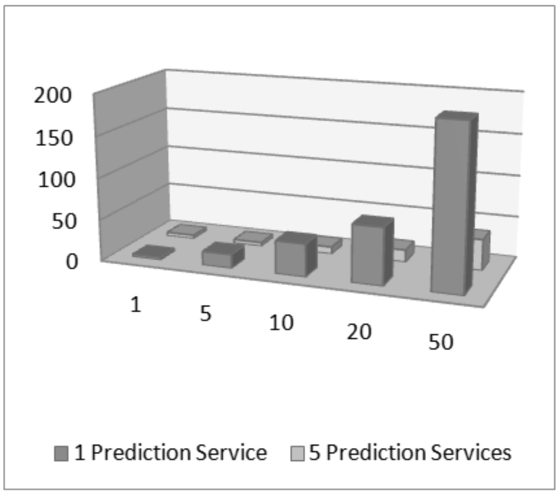

FIGURE 10. Average time needed to generate all solutions.

with only one Prediction Service running in the same workstation on which the system was running. Then, five Prediction Services were replicated also in the same workstation. For every new test, the case base of the PGS was deleted in order to avoid a learning capability, thus requiring the service to accomplish the entire prediction process.

Figure 10 shows the average time needed by OSM for generating all solutions (predictions) for each test group. As can be seen, the time exponentially increases when there is only one PGS running. This is because the service must finish a request to start the next one. So, for the last test group (50 data sets) the service was overcharged. On the other hand, with five replicated services, the system can distribute the requests among these services and optimize the overall performance. The system performed slightly faster when processing a single request, but the performance was constantly reduced when more requests were sent to the service.

Figure 11 shows the average number of crashed agents and services during all tests. As can be seen, with only one PGS available OSM is far more unstable. This is because the PGS had to perform all requests by itself. It is important to notice that when the PGS crashed, more agents crashed because they were always waiting for the service response. For example, when processing 50 data sets, the last agent had to wait almost 200 seconds to receive the response. These data demonstrate that a distributed approach provides a higher ability to recover from errors.

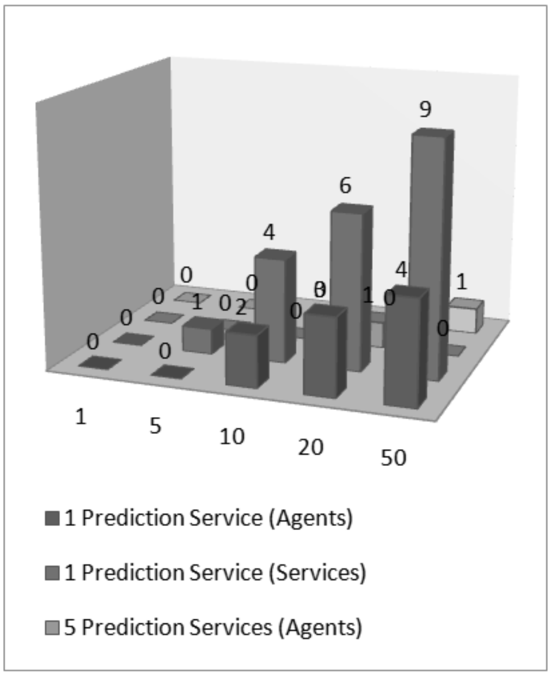

Figure 11. Average number of crashed agents.

5. Conclusions and future work

In this paper, the OSM system has been explained. It is a new solution for predicting the presence or not of oil slicks in a certain area after an oil spill.
OSM represents a combination of a distributed multi-agent architecture with a predicting system based on Case-Based Reasoning. The arrangement of those two methodologies allows OSM to be able to interact with different users at the same time, generating solutions to new problems using past solutions given to past problems.

The distribution has been effective, permitting the users to interact with the system in different ways, depending on the needs of the users or on the requirements of the system.

CBR represents the predicting part of the system. It has proved to generate consistent results if enough data is available. The structure of the CBR methodology has been divided into services in order to adapt its way of working to the inner structure of the multi-agent architecture.

Generalization must be done in order to improve the system. Applying the methodology explained before to diverse geographical areas will make the results even better, being able to generate good solutions in more different situations.

Although these tests have provided us with very useful data, it is necessary to continue developing and enhancing OSM. Results also demonstrate that using a distributed architecture is adequate for building complex systems and exploiting composite services, in this case OSM.

Acknowledgment
This work has been supported by the Spanish Ministry of Science and Technology project TIN2006- 14630-C03-03 and the JCyL project SA071A08.

References

[1] Aamodt, A. and Plaza, E., *Case-Based Reasoning: foundational Issues*, Methodological Variations, and System Approaches, AI Communications, **7** (1) (1994) , 39-59.

[2] Bratman, M.E., *Intentions, plans and practical reason*, Harvard University Press, Cambridge, MA, USA.(1987)

[3] Bratman, M.E., Israel, D. and Pollack, M.E., *Plans and resource-bounded practical reasoning*, Computational Intelligence, **4** (1988) , 349-35.

[4] Brovchenko, I., Kuschan, A., Maderich, V. and Zheleznyak, M., *The modelling system for simulation of the oil spills in the Black Sea*, 3rd EuroGOOS Conference: Building the European capacity in operational oceanography., **192** (2002) .

[5] Cerami, E., *Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL*, O'Reilly & Associates, Inc.

[6] Corchado, J.M., Bajo, J., De Paz, Y. and Tapia, D.I., *Intelligent Environment for Monitoring Alzheimer Patients*, Agent Technology for Health Care, Decision Support Systems, In Press.

[7] Corchado, J.M. and Fdez-Riverola, F., *FSfRT: Forecasting System for Red Tides, Applied Intelligence*, **21** (2004), 251-264.

[8] Chen, Y.M. and Wang, S.C. , *Framework of agent-based intelligence system with two-stage decision-making process for distributed dynamic scheduling*,Applied Soft Computing Journal, **7** (1) (2007), 229-245.

[9] Dunteman, G.H., *Principal Components Analysis.* Newbury Park, California (1989).

[10] Fritzke, B., *Growing cell structures-a self-organizing network for unsupervised and supervised learning*, Neural Networks, **7** (9) (1994), 1441-1460.

[11] Georgeff, M. and Rao, A. , *Rational software agents: from theory to practice*, In Jennings, N.R. y Wooldridge, M.J. (eds), Agent Technology: Foundations, Applications, and Markets. Springer-Verlag New York, Secaucus, NJ, (1998), 139-160.

[12] Gunter, S., Schraudolph, N.N. and Vishwanathan, S.V.N., *Fast Iterative Kernel Principal Component Analysis*, Journal of Machine Learning Research, **8** (2007), 1893-1918.

[13] Haykin, S.,*Neural networks.* Prentice Hall Upper Saddle River, NJ (1999) .

[14] Jayaputera, G.T., Zaslavsky, A.B. and Loke, S.W.,*Enabling run-time composition and support for heterogeneous pervasive multi-agent systems.* Journal of Systems and Software,**80**(12) (2007), 2039-2062.

[15] Jennings, N.R. and Wooldridge, M.,*Applying agent technology.* Applied Artificial Intelligence,**9**(4) (1995), 351-361.

[16] Karayiannis, N.B. and Mi, G.W.,*Growing radial basis neural networks: merging supervised andunsupervised learning with network growth techniques*, Neural Networks, IEEE Transactions on, **8** (6) (1997) , 1492-1506.

[17] Menemenlis, D., Hill, C., Adcroft, A., Campin, J.M., et al. , *NASA Supercomputer Improves Prospects for Ocean Climate Research*, EOS Transactions, **86** (9) (2005) , 89-95.

[18] Palenzuela, J.M.T., Vilas, L.G. and Cuadrado, M.S., *Use of ASAR images to study the evolution of the Prestige oil spill off the Galician coast*, International Journal of Remote Sensing, **27** (2006) (10), 1931-1950.

[19] Plaza, E., Armengol, E. and Ontañón, S.,*The Explanatory Power of Symbolic Similarity in Case-Based Reasoning*, Artificial Intelligence Review, **24** (2) (2005) , 145-161.

[20] Pokahr, A., Braubach, L. and Lamersdorf, W.,*Jadex: Implementing a BDI-Infrastructure for JADE Agents*, In EXP - in search of innovation (Special Issue on JADE) (2003) , 76-85.

[21] Preisendorfer, R.W.,*Principal Component Analysis in Meteorology and Oceanography*, Development in atmospheric science(1988).

[22] Rao, A.S. and Georgeff, M.P.,*BDI agents: from theory to practice*, In Proceedings of the First International Conference on Multi-Agents Systems (ICMAS'95). San Francisco, CA, USA. (1995).

[23] Ros, F., Pintore, M. and Chrétien, J.R., *Automatic design of growing radial basis function neural networks based on neighboorhood concepts*, Chemometrics and Intelligent Laboratory Systems, **87** (2) (2007), 231-240.

[24] Ross, B.J., Gualtieri, A.G., Fueten, F., Budkewitsch, P., et al., *Hyperspectral image analysis using genetic programming*, Applied Soft Computing, **5**(2) (2005), 147-156.

[25] Schn, B., O'Hare, G.M.P., Duffy, B.R., Martin, A.N., et al., *Agent Assistance for 3D World Navigation*, Lecture Notes in Computer Science, **3661**(2005), 499-499.

[26] Snidaro, L. and Foresti, G.L., *Knowledge representation for ambient security*, Expert Systems, **24**(5) (2007), 321-333.

[27] Solberg, A.H.S., Storvik, G., Solberg, R. and Volden, E., *Automatic detection of oil spills in ERS SAR images*, IEEE Transactions on Geoscience and Remote Sensing, **37** (4) (1999), 1916-1924.

[28] Vethamony, P., Sudheesh, K., Babu, M.T., Jayakumar, S., et al., *Trajectory of an oil spill off Goa, eastern Arabian Sea: Field observations and simulations*, Environmental Pollution (2007).

[29] Wooldridge, M. and Jennings, N.R., *Intelligent Agents: Theory and Practice*, The Knowledge Engineering Review,**10**(2) (1995), 115-152.

[30] Yang, J. and Luo, Z., *Coalition formation mechanism in multi-agent systems based on genetic algorithms*, Applied Soft Computing Journal,**7**(2) (2007), 561-568.

Juan Manuel Corchado
Department of Computing Science and Automatic
Plaza de la Merced, s/n
Salamanca
Spain
e-mail: `corchado@usal.es`

Aitor Mata
Department of Computing Science and Automatic
Plaza de la Merced, s/n
Salamanca
Spain
e-mail: `aitor@usal.es`

Sara Rodríguez
Department of Computing Science and Automatic
Plaza de la Merced, s/n
Salamanca
Spain
e-mail: `srg@usal.es`