Applied Soft Computing 11 (2011) 3147–3159

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

CROS: A Contingency Response multi-agent system for Oil Spills situations

Aitor Mata*, Juan M. Corchado, Dante I. Tapia

Department of Computer Science, University of Salamanca, Plaza de la Merced s/n, Salamanca, Spain

ARTICLE INFO

Article history:
Received 10 October 2008
Received in revised form 12 July 2010
Accepted 12 December 2010
Available online 30 December 2010

Keywords:
Oil Spill
Multi-agent systems
Case-Based Reasoning
Service Oriented Architectures

ABSTRACT

This paper presents CROS, a Contingency Response multi-agent system for Oil Spill situations. The system uses the Case-Based Reasoning methodology to generate predictions to determine the probability of finding oil slicks in certain areas of the ocean. CBR uses past information to generate new solutions to the current problem. The system employs a SOA-based multi-agent architecture so that the main components of the system can be remotely accessed. Therefore, all functionalities (applications and services) can communicate in a distributed way, even from mobile devices. The core of the system is a group of deliberative agents acting as controllers and administrators for all applications and services. CROS manages information such as sea salinity, sea temperature, wind speed, ocean currents and atmosphere pressure, obtained from several sources, including satellite images. The system has been trained using historical data obtained after the Prestige accident on the Galician west coast of Spain. Results have demonstrated that the system can accurately predict the presence of oil slicks in determined zones after an oil spill. The use of a distributed multi-agent architecture has been shown to enhance the overall performance of the system.

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 task assignation when organizing resources, especially when multiple people and systems are involved.

This paper presents CROS (*Contingency Response multi-agent system for Oil Spill situations*), a system for helping to manage these situations. This system deploys a prediction model which uses intelligent agents and Case-Based Reasoning systems to determine the possibility of finding oil slicks in a certain area of the ocean. It also applies a distributed multi-agent architecture based on *Service Oriented Architectures* (SOA), modeling most of the system's functionalities as independent applications and services. These functionalities are invoked by deliberative agents acting as coordinators.

Agents and multi-agent systems have been successfully applied to several scenarios, such as education, culture, entertainment, medicine and robotics [1–6]. Agents have a set of characteristics, such as autonomy, reasoning, reactivity, social abilities, pro-activity, mobility and organization which allow them to cover several needs for developing contingency response systems [7].

The agents' characteristics make them appropriate for developing dynamic and distributed systems, as they possess the capability of adapting themselves to the users and environmental characteristics [8]. In addition, the continuous advances in mobile computing make it possible to obtain information about the environment and also to react physically over it in more innovative ways. The agents in CROS multi-agent system are based on the deliberative (*Belief*, *Desire*, *Intention* – BDI) model [9], where the agents' internal structure and capabilities are based on mental aptitudes, using beliefs, desires and intentions for solving problems. However, modern developments need higher adaptation, learning and autonomy levels than pure BDI models [9]. This can be achieved by modeling the agents' characteristics in order to provide them with mechanisms that allow them to solve complex problems and achieve autonomous learning. Some of these mechanisms are *Case-Based Reasoning* (CBR) systems [10], where problems are solved using solutions applied to solve similar past problems [1]. Solutions are stored in a case memory which the system can consult in order to find better solutions for new problems. Deliberative agents can use these systems to learn from past experiences and to adapt their behavior according to each situation.

The system presented in this paper generates the probability of finding oil slicks for different geographical areas after an oil spill. Predictions are created using a Case-Based Reasoning system. The cases used by the CBR system contain information about the oil slicks (size and number) and atmospheric data (wind, ocean currents, salinity, temperature, height and pressure). CROS combines artificial intelligence techniques in order to improve the efficiency of the CBR system, thus generating better results. CROS was trained using historical data acquired during the Prestige oil spill on the Galician west coast of Spain, from November 2002 to April 2003.

* Corresponding author. Tel.: +34 923294400; fax: +34 923294514.
E-mail address: aitor@usal.es (A. Mata).

1568-4946/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2010.12.017

Most of the data used by CROS was acquired from the ECCO (Estimating the Circulation and Climate of the Ocean) consortium [11]. The position and size of the slicks was obtained by treating SAR (*Synthetic Aperture Radar*) satellite images [12]. The development of agents is an essential component in the analysis of data from distributed sensors and provides those sensors with the ability to work together and analyze complex situations [13].

The excessive centralization of services negatively affects the systems' functionalities, overcharging or limiting their capabilities. Classical functional architectures are characterized by trying to find modularity and a structure oriented to the system itself. Modern functional architectures like *Service-Oriented Architecture* (SOA) consider integration and performance aspects that must be taken into account when functionalities are created outside the system.

As described by [14], "A SOA-based system is a network of independent services, machines, the people who operate, affect, use, and govern those services as well as the suppliers of equipment and personnel to these people and services". The term service can be defined as a mechanism that facilitates the access to one or more functionalities (e.g. functions and network capabilities). Services are linked by means of standard communication protocols that must be used by applications in order to share resources in the services network. A SOA approach has been chosen in CROS because such architectures are asynchronous and non-dependent on context (i.e. previous states of the system) [15].

Agents and multi-agent systems combine classical and modern functional architectural aspects. Multi-agent systems are structured by taking into account the modularity in the system, and by reuse, integration and performance of the data and results obtained. CROS employs a SOA-based multi-agent architecture which allows different users to work together without sharing the same space. The architecture divides the main components of the system into small pieces of software which work separately but are coordinated. These components are managed through a set of deliberative BDI agents. The use of a distributed architecture can help to distribute resources and reduce the central unit tasks [16,17]. It also provides more flexible ways of moving functions to where actions are needed, thus obtaining better responses at execution time, autonomy, services continuity, and superior levels of flexibility and scalability than centralized architectures [18].

The following section presents the oil spill problem while highlighting the difficulties and the possibilities of finding solutions to this problem. This is followed by a brief description of the CBR methodology and the systems created from it. Afterwards, the main components of the system presented in this study, including its architecture, are described. Finally, the results obtained by applying CROS to a real oil spill situation, the final conclusions and future work are presented.

2. Facing the Oil Spill problem

Predicting the behavior of oceanic elements is quite a difficult task. In this case, the prediction is related to external elements (oil slicks), making the prediction even more difficult. The open ocean is a highly complex system that can be modeled by measuring different variables and structuring them together. Some of these variables are essential in predicting the behavior of oil slicks. It is necessary to know the previous positions of oil slicks in order to predict the future presence in a specific area. That knowledge is provided by the analysis of satellite images which reveal the precise position of the slicks.

It is very important to determine if an area will be contaminated or not after an oil spill. It is necessary to know how the slicks generated by the spill behave for concluding about the presence of contamination in a specific area. First, the position, shape and size of the oil slicks must be identified. One of the most precise ways to acquire that information is using satellite images. SAR images are the most commonly used to automatically detect these kinds of slicks [19]. Satellite images show certain areas where there seems to be nothing (e.g. zones with no waves), such as oil with slicks. Fig. 1 (top) shows a SAR image which displays a portion of the Galician west coast with black areas corresponding to oil slicks.

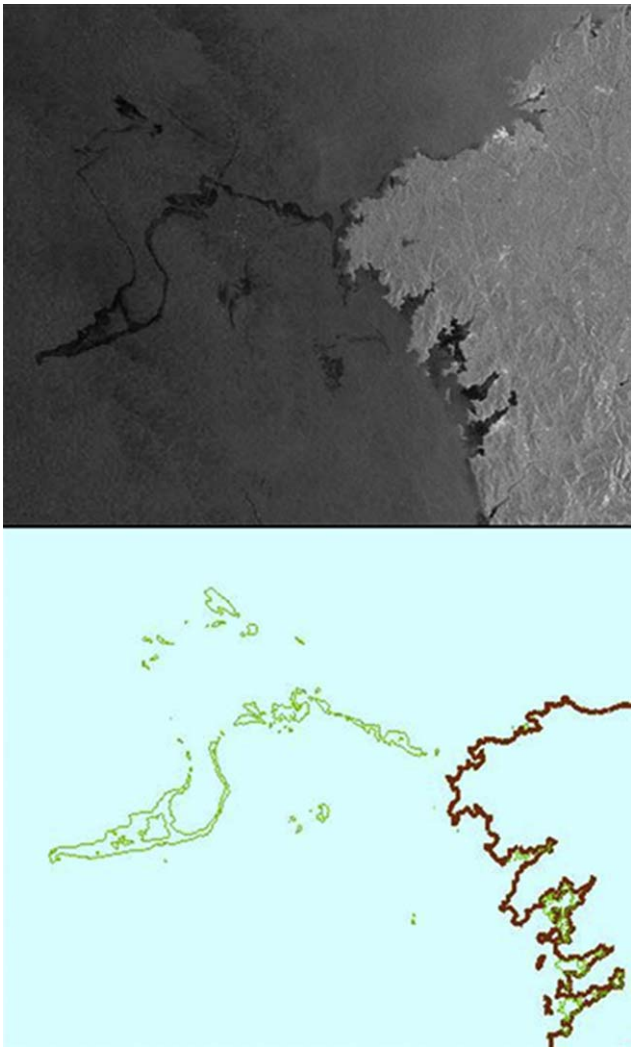

Fig. 1. . Satellite image of an oil spill near the Galician west coast in Spain (top) and its interpretation by the CROS system (bottom).

Fig. 1(bottom) also shows the interpretation of the SAR image done by CROS 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 calm 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 of this problem that can be automatically managed by computational tools [20]. 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 it 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 collected in order to obtain a global model that can explain how slicks evolve.

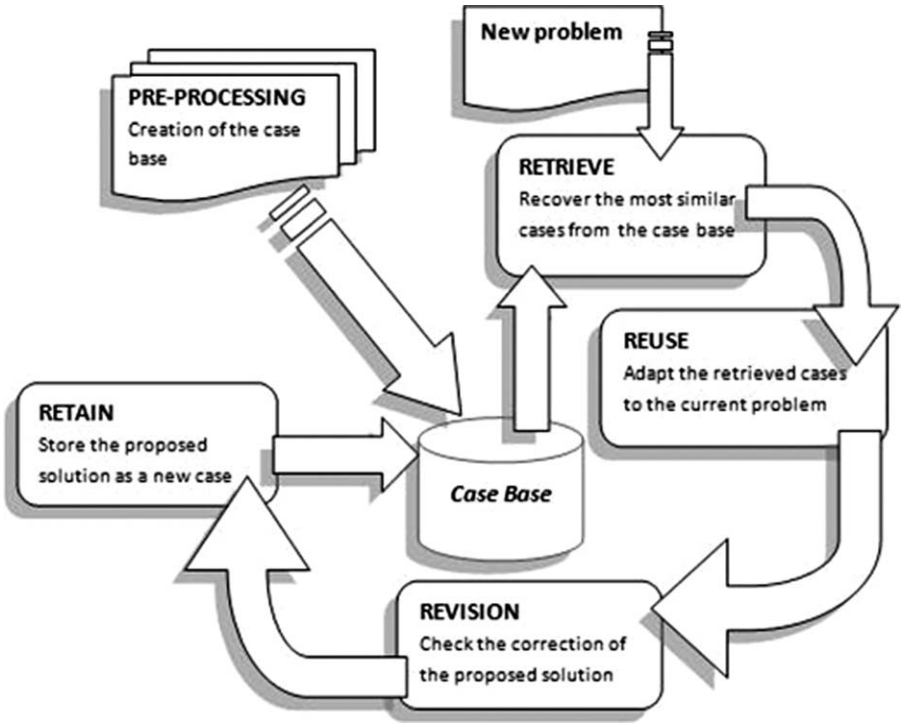

Fig. 2. . Basic structure of a CBR system.

There are different ways to analyze, evaluate and predict situations after an oil spill. One approach is simulation [21], where the model of a certain area is created by introducing specific parameters (weather, currents and wind) and working together with a forecasting system. Using simulations it is easy to obtain a good solution for a certain area. However, it is quite difficult to generalize this information in order to solve the same problem in related areas or new zones. It is also possible to replace the oil spill with drifters [22] to obtain a trajectory model by comparing the trajectory followed by the drifters with the already known oil slick trajectories. If the drifters follow a trajectory similar to the one that followed the slicks, then a model can be created and there it may be possible to create more models in different areas. Another way of predicting oil slicks trajectories is to study previous cases for obtaining a trajectory model for a certain area [23]. Taking these solutions one step further can be accomplished using systems that combine a major set of elements that generate response models to solve the oil spill problem. A different point of view is given by complex systems that use expert systems to analyze large databases (environmental, ecological, geographical and engineering) [24]. This way, an implicit relationship between the problem and the 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 that enable a quick solution. 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 create a decision support system that will 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 [25].

The next section briefly explains the CBR methodology, focusing on the possibility of creating systems intended to solve different situations.

3. Case-Based Reasoning systems

Case-Based Reasoning (CBR) is a methodology whose capabilities are based on past experiences. It has its origin in knowledge based systems and has been used in several scenarios such as health care, e-Learning, ubiquitous computing and oceanography [26–28]. CBR systems learn from previous situations. The main element of a CBR system is the case base, which is a structure that stores problems, elements (*cases*) and their solutions. A case base can be visualized as a database where a collection of problems is stored together with the solutions related to each problem. This allows the system to generalize in order to solve new problems, which is precisely the main objective of those systems.

The learning capabilities of CBR are due to its own structure, which is composed of four main phases [10]: *retrieval*, *reuse*, *revision* and *retain*. These phases are represented in Fig. 2, with a brief description of the main aim of each phase.

The *retrieve* phase consists of finding the most similar cases to the current problem from the case base. Once a series of cases are extracted they can be reused. The *reuse* phase adapts the selected cases for solving the current problem through a new solution. The *revision* phase revises the solution to check if it is an adequate solution to the current problem. If the solution is confirmed, then it is *retained* in the retain phase. This new solution could eventually serve as a solution for future problems. In most cases, CBR should not been used alone but combined with artificial intelligence techniques in each phase. For example, Growing Cell Structures have been used with CBR to automatically create the internal structure of the case base from existing data, and they has been combined with multi-agent applications [29] to improve their results. ART-Kohonen neural networks, artificial neural networks, genetic algorithms and fuzzy logic [30,31] have also been used to complement the capabilities of CBR systems. These techniques enhance the way CBR systems generate better solutions.

The main functionalities of CROS will now be described, including the internal CBR system used for generating predictions

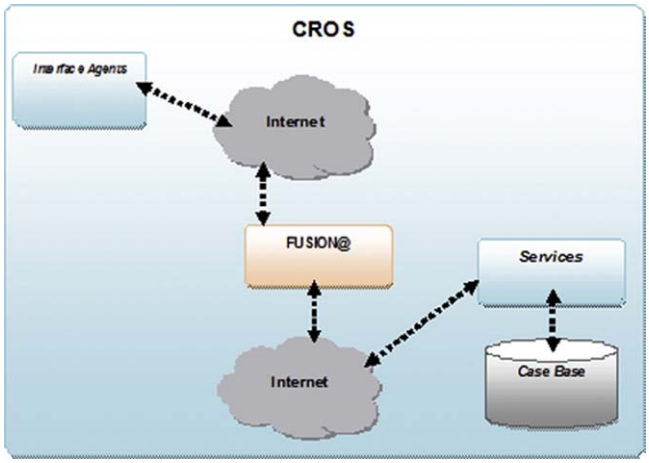

Fig. 3. . Communication architecture in CROS.

regarding the presence of oil slicks in oceanic zones, and the different agents involved in the system.

4. CROS: a Contingency Response multi-agent system for Oil Spills

CBR systems have already been applied to solve maritime problems [26] in which different oceanic variables were involved. In CROS, the data collected from different satellites are processed and structured as cases. Cases are the key to obtaining solutions to future problems through a CBR system. The functionalities of CROS can be accessed using different interfaces executed on PCs or PDAs (Personal Digital Assistants). Users can interact with the system by introducing data, requesting a prediction or revising a generated solution (i.e. prediction). Fig. 3 shows the basic communication schema among the various components of CROS. As shown, the interface agents communicate with the services through the agent platform (e.g. JADE, JACK and OAA) and vice versa.

Fig. 4 shows the main graphical user interface (GUI) of CROS. The GUI shows a set of parameters, the oceanic area visualization with oil slicks, and the squared area to be analyzed. The interface agents perform all the different functionalities that users can use to interact with CROS. The different phases of the CBR system have been modeled as services, so each phase can be requested independently. For example, one user may only input information into the system (e.g. a new case), while another user could request a new prediction.

Oil slicks are mainly detected by SAR images. Those images are processed and transformed to be used by the system. Oceanic, meteorological and oil spill related data are stored in the system in order to generate future predictions. The data used to train the system was obtained after the Prestige accident, between November 2002 and April 2003, in a specific geographical area on the western coast of Galicia (longitude between 14° and 6° west and latitude between 42° and 46° 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), or directly related to the problem (number and area of the slicks).

The variables stored in the case base were chosen according to the explanations given by experts in oceanic evolution. The variables shown in Table 1 represent the necessary information to eventually predict the evolution of an oil slick in open ocean. The *area of the slicks* parameter in the case base represents the parameter that is going to be predicted. That prediction will be based on the values of the rest of the parameters (e.g. latitude, longitude, sea height and temperature) which are the inputs of the neural network that will generate the future value of the *area of the slicks* parameter, that is, the solution to the problem. The future situation of the *area of the slicks* parameter will be generated from the rest of the parameters of the problem introduced in the system and also of the cases retrieved from the case base.

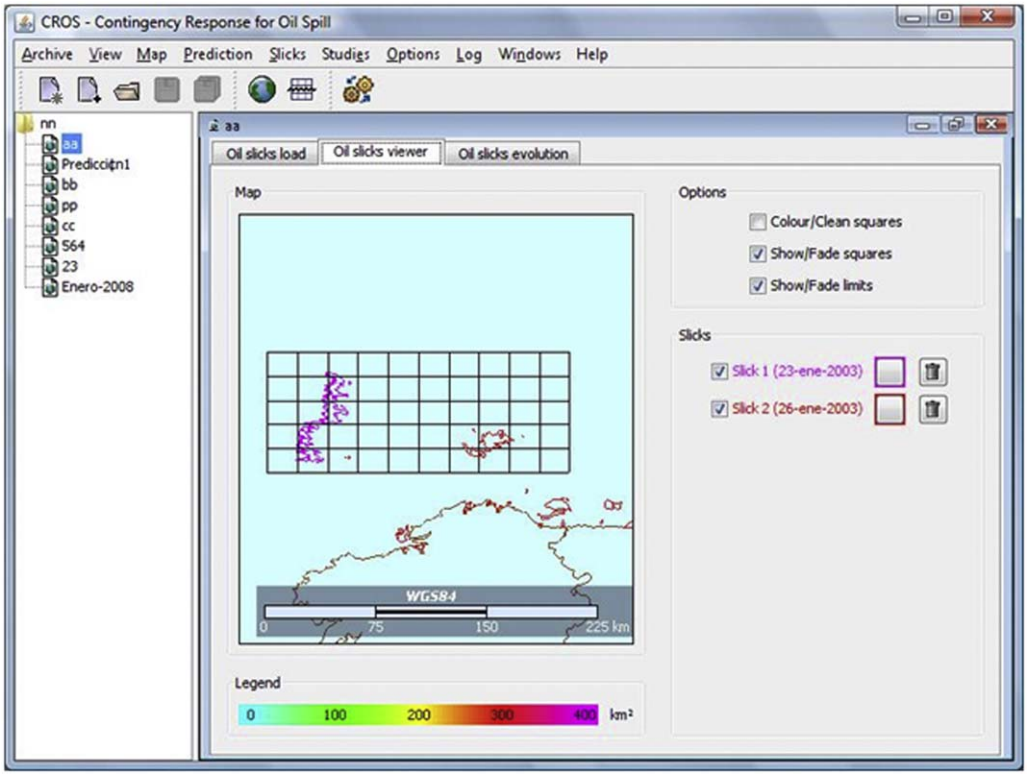

Fig. 4. . Graphical user interface of CROS.

Table 1
Variables that define a case.

Variable	Definition	Unit
Longitude	Geographical longitude	◦
Latitude	Geographical latitude	◦
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	N/m^2
Salinity	Sea salinity	ppt (parts per thousand)
Temperature	Celsius temperature in the area	◦C
Meridional wind	Meridional direction of the wind	m/s
Zonal wind	Zonal direction of the wind	m/s
Wind strength	Wind strength	m/s
Meridional current	Meridional direction of the ocean current	m/s
Zonal current	Zonal direction of the ocean current	m/s
Current strength	Ocean current strength	m/s
Area of the slicks	Surface covered by the slicks present in the analyzed area	km^2

All the available information is stored in the case base, after which CROS is ready to predict future situations. The results obtained by CROS will specifically depend on the quality and quantity of information stored in the case base. A problem situation must be introduced in the system in order to generate a prediction. Then, the most similar cases to the current situation are retrieved from the case base. Once a group of cases is chosen from the case base, they must be used to generate a new solution to the current problem. Growing Radial Basis Functions Networks [32] are used in CROS to combine the chosen cases and obtain the new solution.

CROS determines the probability of finding oil slicks in a certain area. CROS divides the area to be analyzed in squares of approximately half a degree per side to generate the new prediction. The system then determines the amount of slicks in each square. The squares are colored with different gradations depending on the quantity of oil slicks calculated.

Within the case base there is a temporal relationship between a case and its future location. A square containing all the values of the different variables can be related to the same square in the next temporal location. That relationship will provide the internal mechanism used to generalize and to train the GRBF network, which will generate the prediction. Because every case is related to its *next future location* stored in the case base, it is easy to know the evolution of every single element within the case base and, thus, easier to generalize the evolution of the slicks in different situations.

Fig. 5 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 (top). Then, the squared zones are colored in different intensities depending on the number of slicks appearing in each square (bottom). The larger the number of slicks, the darker the square is colored.

Once the available data is structured, it is stored in the case base. Every case has its temporal situation stored and every case is related to 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 a newly introduced problem.

4.1. CROS main architecture

CROS 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 [9,33]. 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 [34].

There are four basic blocks in CROS: Applications, Services, Agent 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,

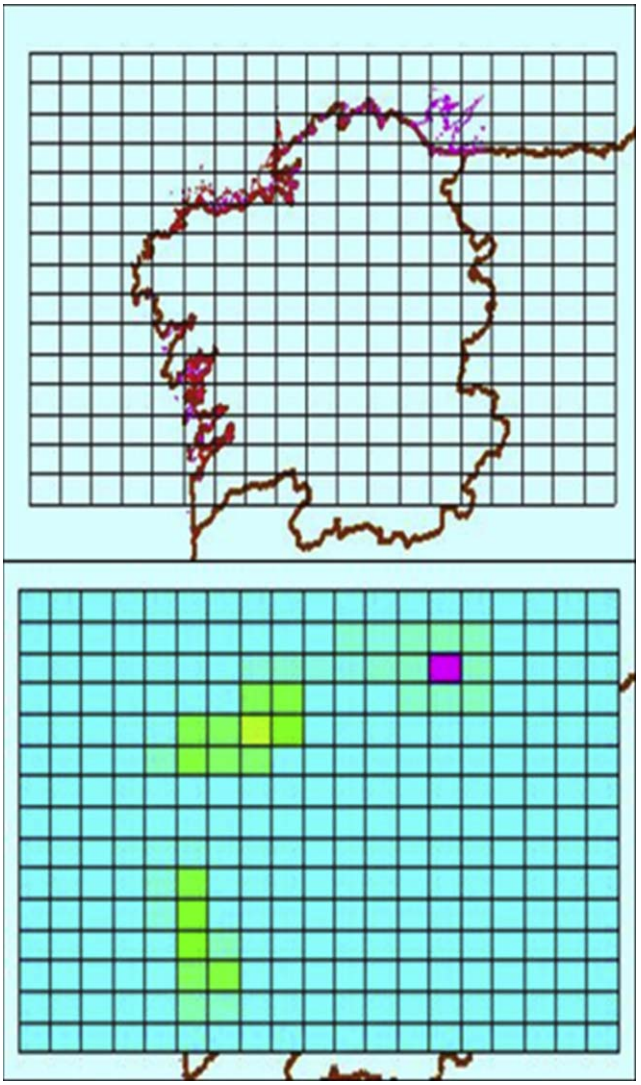

Fig. 5. . Division of areas into squares and colors applied to each square.

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. In CROS, the applications are related to the different possibilities that the users have to relate interact with the system. Therefore, there are applications to introduce information, to request a prediction or to revise an automatic solution.

- ***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. CROS 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. CROS internal services cover the main phases of the CBR methodology, allowing the applications to introduce information in the case base, to request a prediction and to consult the information available in the case base.
- ***Agent 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 CROS, 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 Agent 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 [35]. Services and applications communicate with the *Agent Platform* via SOAP messages, such as the request defined below. 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 CROS, so while the communication protocol is not needed in all instances, it is absolutely required for all services.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12
/soap-envelope"
soap:encodingStyle="http://www.w3.org
/2001/12/soap-encoding">

<soap:Body
xmlns:m="http://www.example.org/spill
">
  <m:GetSpillInfo>
    <m:StockSize>IBM</m:StockSize>
  </m:GetStockInfo>
</soap:Body>

</soap:Envelope>
```

Agents, applications and services in CROS 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 the resources, so the system has an elevated adaptability and capacity for error recovery. Users can access CROS functionalities through distributed applications which run on different types of devices and interfaces (e.g. computers, PDA). Fig. 6 shows a more detailed structure of CROS. As shown, 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 to multiple requests.

Interface Agents are a special kind of agent in CROS designed to be embedded in user 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. CROS defines three different *Interface Agents*:

1. *Input Agent*. This agent sends the information introduced by the users to CROS. Once the data have reached the system, it is structured into the case base. This interface agent is employed by users that have visualized an oil slick, and then introduce the data related to that slick into the system. Fig. 7 shows the *Input Agent* interface shown to the users who 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, CROS 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 CROS, it uses this agent to do so. With the agent interface, shown in Fig. 8, the user can define the area to be analyzed, the size of the squares to be transformed into cases and, if there is 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 CROS, 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 receive the proposed solution and enough data to validate the solution for the current problem. Fig. 9 shows the interface with which the expert user can verify the correction of the solution proposed.

CROS also defines three different services that 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 CROS. The internal CBR system used to generate the predictions and administrate the case base in CROS will now be explained. The main services that form that system will be explained in greater detail.

4.2. Internal CBR system

CROS is a Contingency Response multi-agent system for Oil Spills that uses the Case-Based Reasoning methodology by implementing an internal CBR system that deals with the information available in the case base. The CBR system created to solve CROS information needs to be divided into independent and distributed services which cover the four main phases of the CBR cycle. The *Prediction Generation Service* combines the retrieval and reuse phases.

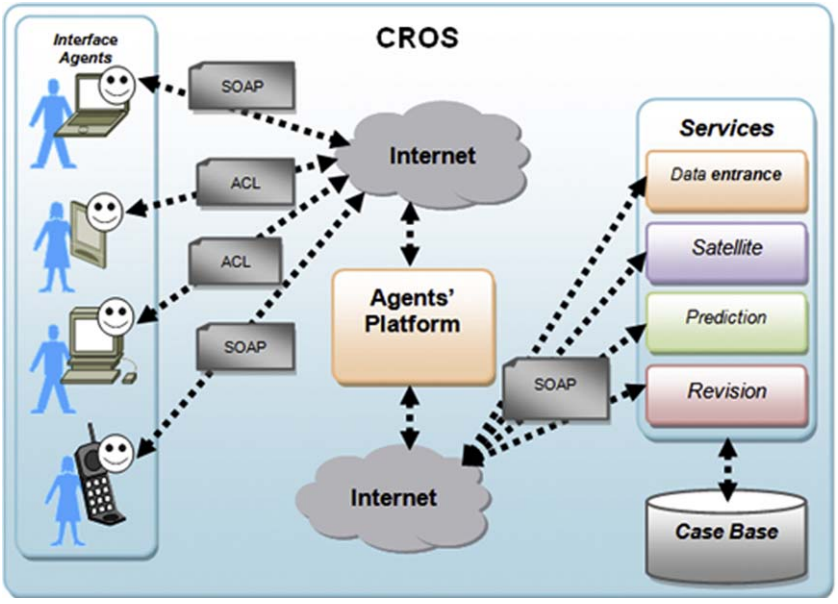

Fig. 6. . CROS structure.

This service generates a prediction after a problem description is introduced in the system (i.e. in CROS). The *Revision Service* covers the revision phase. It requires a confirmation of an expert (user) for validating the solution proposed. Finally, the *Data Input Service* integrates the retention phase, closing the whole CBR cycle. These three services will now be described.

4.2.1. Data Input Service

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

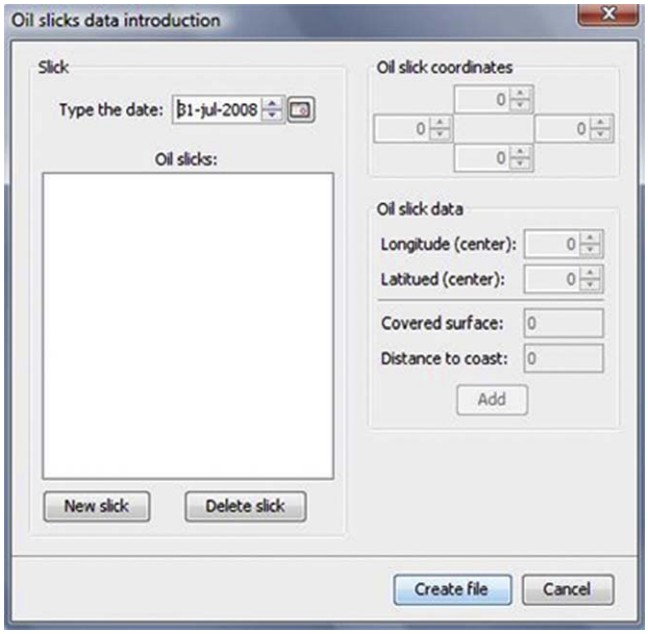

Fig. 7. . Input Agent interface.

Historical data collected from November 2002 to April 2003 were used to create the initial case base for CROS. As previously explained, cases are formed by a series of variables. Principal Components Analysis (PCA) [36] can reduce the number of those variables, after which the system stores the value of the principal components, which are then related to 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 [37]. Some adaptations and modifications of the classic PCA algorithm has been recently used to solve different kinds of problems, and applied to quite different scenarios [38–42]. CROS uses Fast Iterative Kernel PCA (FIKPCA), which is an evolution of PCA [43]. 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 previous Kernel Hebbian Algorithm used by classic Kernel PCA, FIK-PCA set η_t (the appropriate scalar gain) proportional to the reciprocal of the estimated values. Let $\lambda_t \in \Re_+^r$ denote the vector of values associated with the current estimate of the first r eigenvectors. The new KHA algorithm sets the ith component of η_t to the files. τ is positive tuning parameters. τ determines the length of an initial search phase with near-constant gain, with η_0 and τ being positive tuning parameters. Since the working phase is not intended to start before all observations have been seen at least once, τ is tuned into small integer multiples of the data set size l.

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

The final variables are linearly independent and are formed by a combination of previous variables. The values of the original variables can be recovered by applying an inverse calculation to the one produced to obtain the new variables. The least used variables among the final stored variables are those whose values suffer the fewest changes during the periods of time analyzed. Salinity, temperature and pressure do not change from one day to another, so 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, which have the following distribution:

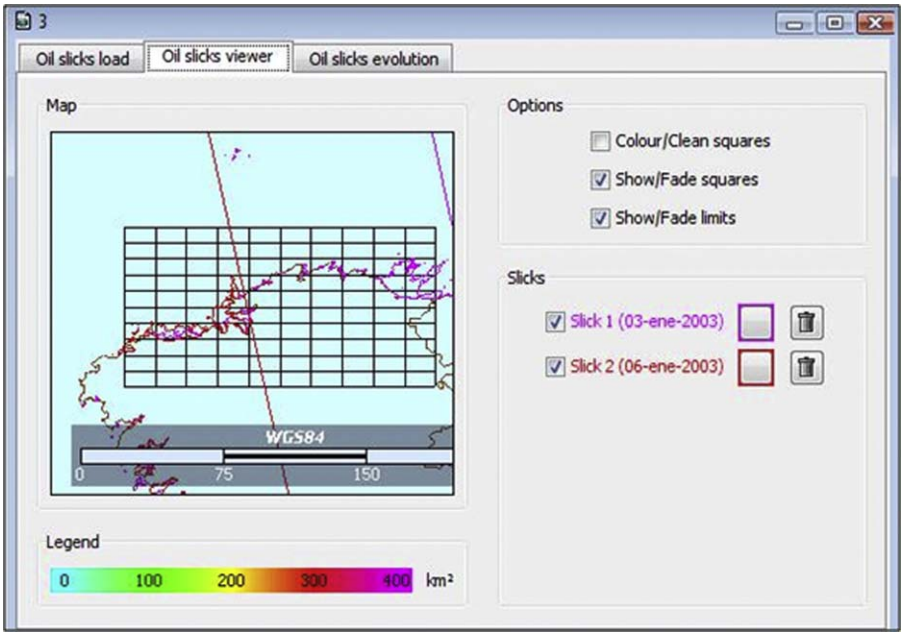

Fig. 8. . Prediction Agent interface.

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

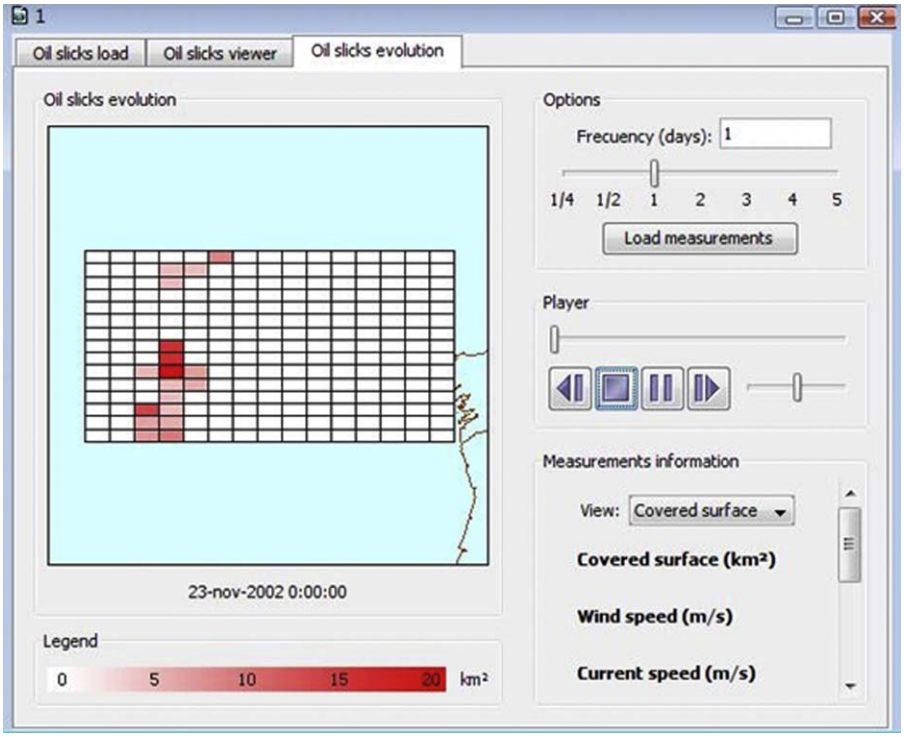

Fig. 9. . *Revision Agent* interface.

the information in about a 60% of the cases, 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) [44] 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 each 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.

$$\omega_c(t+1) = \omega_c(t) + \varepsilon_c(x - \omega_c) \tag{2}$$

$$\omega_n(t+1) = \omega_n(t) + \varepsilon_n(x - \omega_n) \tag{3}$$

The GCS insertion works as follows. First, the most similar cell to the new one is found. The new cell is introduced in the middle of the connection between the most similar cell and the least similar to the new one. Then, direct neighbours of the closest cell change their values by approximating to the new cell and the 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 does the GCS change again, introducing the proposed solution if it is correct.

4.2.2. 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 the 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 to 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 is used to obtain that prediction [45]. The chosen cases are used to train the artificial neural network. Radial Basis Function networks were chosen because of the reduced training time compared to 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 [46] 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, while 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 (4).

$$e_i = \frac{l}{p^*} \sum_{j=1}^{p} ||t_{ij} - y_{ij}||, \tag{4}$$

where t_{ij} is the desired value of the jth output unit of the ith training pattern, y_{ij} the actual values of the jth output unit of the ith training pattern.

The Growing RBF process is described next:

1. Calculate the error, e_i (4) for every new possible prototype:
   (a) If the new candidate does not form part of the selected cases and the error calculated is less than the 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 error threshold, 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 centred on it.
   b. Else, increase the iteration factor. If the iteration factor reaches 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 proposed is the output of the GRBF network created with the retrieved cases. The input to the GRBF network, which is used to generate the solution, is the data related to the problem to be solved, i.e., and the values of the variables stored in the case base.

4.2.3. 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 to how the slicks are interpreted by CROS. A set of squared colored areas appear. The intensity of the color corresponds to 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 who confirms the automatic revision.

Explanations are a recent revision methodology used to check the correction of the solutions proposed by CBR systems [47]. 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 under the given 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 revision process is specified:

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 that of the selected cases is below a certain threshold value, then the solution is considered to be valid.
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 accepting the best generated solution.

The distances are calculated considering the sign of the values without using its absolute value. This decision is justified by the fact that it is not the same to move to the north as 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 be used in future 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.

5. Results

CROS 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, achieving about a 60% 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, the Growing Radial Basis Function Networks was 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 from using Growing RBF networks instead of simple RBF networks are about 4% more accurate, which is a good improvement. Using GRBF also implies a better adaptation to the retrieved data which implies a more coherent solution generation.

Evaluations show that the system can make predictions in known conditions, 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 results than using the CBR alone (17% better), and also better than using the isolated techniques (neural networks), without the integration feature produced by the CBR (11% better). A summary of all these improvements can be seen in Fig. 10.

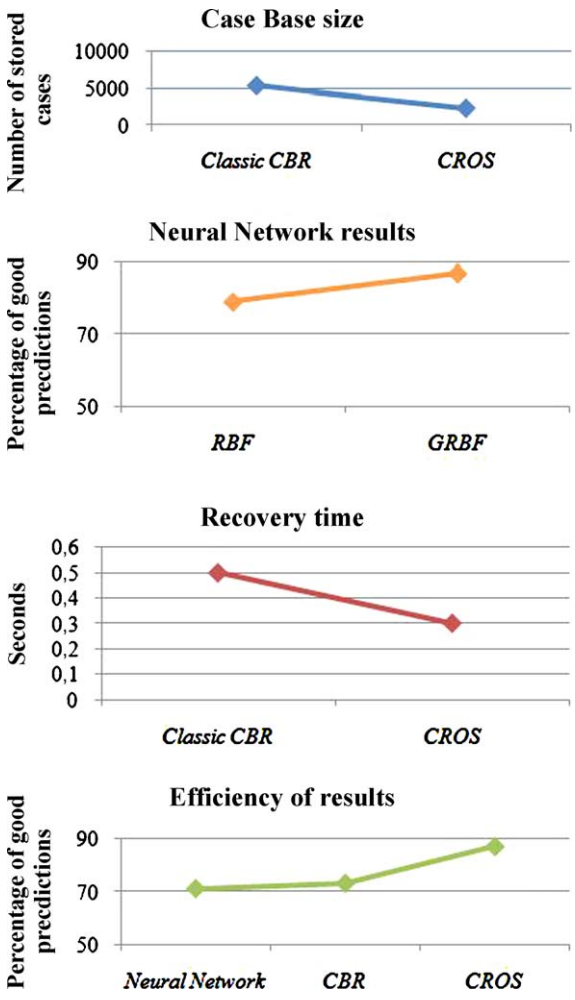

Fig. 10. . Summary of the improvement of the results obtained with CROS.

The predicted situation was compared to the actual future situation. The future situation was known because historical data was used to develop the system and to test its correction. The proposed solution was, for the majority of the variables, close to 90% accurate. 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 neighbours). This type of prediction is used in order to clearly observe the direction of the slicks, which can be useful in determining 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 CROS. The table shows the evolution of the results along with the increase in the number of cases stored in the case base. The results for all of the techniques analyzed improved while increasing the number of cases stored. Having more cases in the case base makes it easier to

Table 2
Percentage of good predictions obtained with different techniques.

Number of cases	RBF	CBR	RBF + CBR	CROS
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 3
Multiple comparison procedure among different techniques.

	RBF	CBR	RBF + CBR	CROS
RBF				
CBR	=			
RBF + CBR	=	=		
CROS	*	*	*	

find cases similar to the proposed problem, which in turn enables the solution to be more accurate. The "*RBF*" column represents a simple Radial Basis Function Network that is trained with all the available data. The network produces an output that is considered a solution to the problem. The network is trained with the majority of the information available and, the rest of the data, which has not been previously used for training purposes, is used to check it. There is no filtering or selection of optimal data. The "*CBR*" column represents a pure CBR system, with no other techniques included; the cases are stored in the case base and recovered considering the Euclidean distance (according to the internal representation of the cases). The most similar cases are selected and, after applying a weighted mean according to the similarity of the selected cases with the inserted problem, a solution s proposed. The values are calculated transforming the information stored into a vector of values, which allows the system to measure the distance between two different elements. The "*RBF + CBR*" column 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 RBF network is now trained with all the information used for training, but when generating solutions, a subset of the data available is introduced in the network to generate the results. The results of the "*RBF + CBR*" column are generally better than those from the "CBR", mainly because of the elimination of useless data to generate the solution. Finally, the "*CROS*" column shows the results obtained by the proposed system, obtaining better results than the three previously analyzed solutions. The laboratory environment where the different techniques were applied was the same for all of them. Three Pentium IV computers were used in the tests (3.00 GHz, 2 Gb RAM, Windows). One of them was used as a server, and the other two were used to send requests and receive the solutions. The same historical data was also used to generate all the results used for comparison.

Table 3 shows a multiple comparison procedure (Mann–Whitney's test) [48] used to determine which models are significantly different from the others. This test helps to determine H_0. That is, whether the variables are identically distributed for two analyzed techniques. In our case, the variables represent the percentage of errors for the numbers of cases shown in Table 2. Therefore, the variables x and y represent the different pairs of techniques (i.e. RBF, CBR, RBF + CBR and CROS) selected. The values for the variables x and y are the successive rows of each column. The asterisk indicates that these pairs show statistically significant differences. We have taken the lowest significance level for H_0 (i.e. $\alpha = 0.13$) where CROS presents statistically significant differences with the rest of the models, while the rest are considered equal. The proposed solution does not generate a trajectory, but a series of probabilities in different areas, which is far more similar to the real behavior of the oil slicks.

Several tests have been done to compare the overall performance of CROS. 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

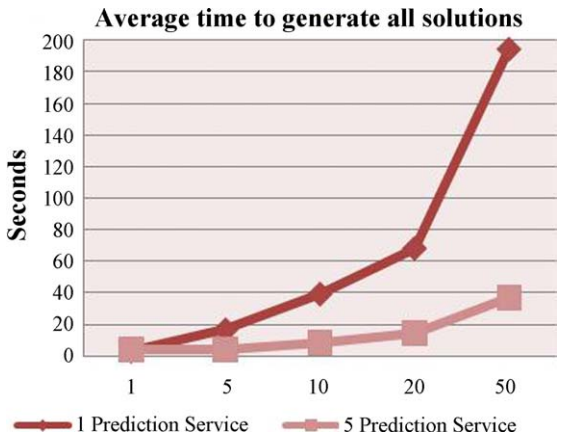

Fig. 11. . Average time needed to generate all solutions.

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 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.

Fig. 11 shows the average time needed by CROS 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.

Fig. 12 shows the average number of crashed agents and services during all tests. As can be seen, with only one PGS available CROS 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,

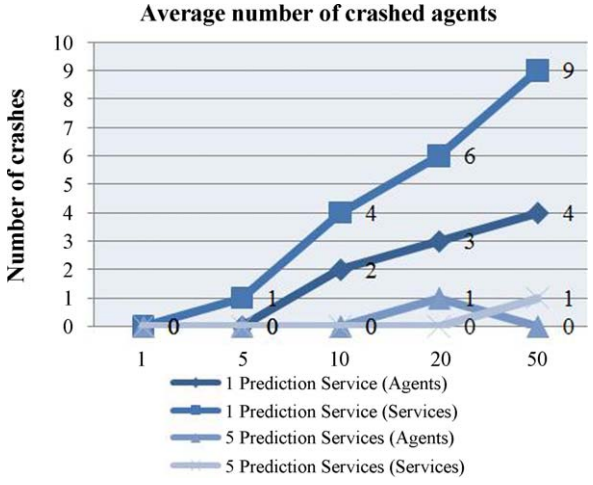

Fig. 12. . Average number of crashed agents.

the last agent had to wait almost 200 s to receive the response. These data demonstrate that a distributed approach provides a higher ability to recover from errors.

6. Conclusions and future work

CROS is a new solution for predicting the presence of oil slicks in oceanic areas after an oil spill. This system has a SOA-based distributed multi-agent architecture which allows the simultaneous interaction of multiple users. Distributing resources also allows users to interact with the system in different ways depending on their specific needs for each situation (e.g. introducing data or requesting a prediction). This distributed and multi-agent architecture is an improvement over previous tools where the information must be centralized and where local interfaces were used. With the vision introduced by CROS, all the different people that may interact with a contingency response system can collaborate in a distributed way, being physically located in different places but exchanging information in a collaborative mode.

CROS uses a Case-Based Reasoning system to create new solutions and predictions using past solutions given to past problems. It has been demonstrated that the CBR system generates consistent results. The structure of the CBR system was divided into services in order to optimize the overall performance of CROS.

In order to improve the system, it will be necessary to incorporate some generalizations. Applying the methodology explained before to diverse geographical areas will improve the results even more, being able to generate good solutions in a larger variety of situations. The current system was mainly developed using data from the accident of the Prestige off the north-west coast of Spain. With that information, CROS was able to generate solutions to new situations, based on the available cases. If the amount and variety of cases stored in the case base increases, the quality of the results will also improve.

Although the performed tests have provided us very useful data, it is necessary to continue developing and enhancing CROS. The number of possible interfaces can be augmented, including independent sensors that may send information to the system in real-time. The data received by the system must be analyzed in order to detect new spills and to generate fast and accurate solutions to existing problems without the direct intervention of the users. Then, the system will be not only a contingency response but also a kind of supervising system that can be specifically used in dangerous geographical areas.

Acknowledgements

This work was supported in part by the projects MEC THOMAS TIN2006-14630-C03-03, IMSERSO 137/07 and JCYL SA071A08.

References

[1] J.M. Corchado, J. Bajo, Y. De Paz, D.I. Tapia, Intelligent environment for monitoring Alzheimer patients, agent technology for health care, Decision Support Systems 44 (2008) 382–396.
[2] B. Schön, G.M.P. O'hare, B.R. Duffy, A.N. Martin, J.F. Bradley, Agent assistance for 3D world navigation, Lecture Notes in Computer Science 3661 (2005) 499–1499.
[3] H.H. Lund, Adaptive robotics in entertainment, Applied Soft Computing Journal 1 (2001) 3–20.
[4] J. Bajo, J.M. Corchado, Y. De Paz, J.F. De Paz, S. Rodríguez, Q. Martín, A. Abraham, SHOMAS: intelligent guidance and suggestions in shopping centres, Applied Soft Computing Journal 9 (2009) 851–862.
[5] M.S. Hamdi, MASACAD: a multi-agent approach to information customization for the purpose of academic advising of students, Applied Soft Computing Journal 7 (2007) 746–771.
[6] G. Cabri, L. Ferrari, L. Leonardi, A role-based mobile-agent approach to support e-democracy, Applied Soft Computing Journal 6 (2005) 85–99.
[7] J. Yang, Z. Luo, Coalition formation mechanism in multi-agent systems based on genetic algorithms, Applied Soft Computing Journal 7 (2007) 561–568.
[8] G.T. Jayaputera, A.B. Zaslavsky, S.W. Loke, Enabling run-time composition and support for heterogeneous pervasive multi-agent systems, Journal of Systems and Software 80 (2007) 2039–2062.
[9] M.E. Bratman, D. Israel, M.E. Pollack, Plans and resource-bounded practical reasoning, Computational Intelligence 4 (1988) 349–355.
[10] A. Aamodt, E. Plaza, Case-based reasoning: foundational issues, methodological variations, and system approaches, AI Communications 7 (1994) 39–59.
[11] D. Menemenlis, C. Hill, A. Adcroft, J.M. Campin, B. Cheng, B. Ciotti, I. Fukumori, P. Heimbach, C. Henze, A. Köhl, NASA supercomputer improves prospects for ocean climate research, EOS Transactions 86 (2005) 89–95.
[12] J.M.T. Palenzuela, L.G. Vilas, M.S. Cuadrado, Use of ASAR images to study the evolution of the Prestige oil spill off the Galician coast, International Journal of Remote Sensing 27 (2006) 1931–1950.
[13] F. Pecora, A. Cesta, Dcop for smart homes: a case study, Computational Intelligence 23 (2007) 395–419.
[14] J.A. Estefan, K. Laskey, F.G. Mccabe, D. Thornton, Reference Architecture for Service Oriented Architecture Version 1.0, 2008 (Online PDF).
[15] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana, Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing 6 (2002) 86–93.
[16] L. Ardissono, G. Petrone, M. Segnan, A conversational approach to the interaction with Web Services, Computational Intelligence 20 (2004) 693–709.
[17] H. Voos, Agent-based distributed resource allocation in technical dynamic systems, in: Proceedings of the IEEE Workshop on Distributed intelligent Systems: Collective intelligence and Its Applications (DIS'06), IEEE Computer Society, Washington, DC, 2006.
[18] L.M. Camarinha-Matos, H. Afsarmanesh, A comprehensive modeling framework for collaborative networked organizations, Journal of Intelligent Manufacturing 18 (2007) 529–542.
[19] A.H.S. Solberg, G. Storvik, R. Solberg, E. Volden, Automatic detection of oil spills in ERS SAR images, IEEE Transactions on Geoscience and Remote Sensing 37 (1999) 1916–1924.
[20] B.J. Ross, A.G. Gualtieri, F. Fueten, P. Budkewitsch, B.J. Ross, A.G. Gualtieri, F. Fueten, P. Budkewitsch, Hyperspectral image analysis using genetic programming, Applied Soft Computing 5 (2005) 147–156.
[21] I. Brovchenko, A. Kuschan, V. Maderich, M. Zheleznyak, The modeling system for simulation of the oil spills in the Black Sea, in: 3rd EuroGOOS Conference: Building the European Capacity in Operational Oceanography, 2002, p. 192.
[22] J.M. Price, Z.G. Ji, M. Reed, C.F. Marshall, M.K. Howard, N.L. Guinasso Jr., W.R. Johnson, G.B. Rainey, Evaluation of an oil spill trajectory model using satellite-tracked, oil-spill-simulating drifters, in: OCEANS 2003. Proceedings, 2003, p. 3.
[23] P. Vethamony, K. Sudheesh, M.T. Babu, S. Jayakumar, R. Manimurali, A.K. Saran, L.H. Sharma, B. Rajan, M. Srivastava, Trajectory of an oil spill off Goa, eastern Arabian Sea: field observations and simulations, Environmental Pollution (2007) 148.
[24] C. Douligeris, J. Collins, E. Iakovou, P. Sun, R. Riggs, C.N.K. Mooers, Development of OSIMS: an oil spill information management system, Spill Science & Technology Bulletin 2 (1995) 255–263.
[25] X. Liu, K.W. Wirtz, Sequential negotiation in multiagent systems for oil spill response decision-making, Marine Pollution Bulletin 50 (2005) 469–474.
[26] J.M. Corchado, F. Fdez-Riverola, FSfRT: forecasting system for red tides, Applied Intelligence 21 (2004) 251–264.
[27] I. Bichindaritz, Memory organization as the missing link between case-based reasoning and information retrieval in biomedicine, Computational Intelligence 22 (2006) 148–160.
[28] N. Pincho, V. Marques, A. Brito, J.T. Farinha, E-learning by experience-how CBR can help, WSEAS Transactions on Advances in Engineering Education 3 (2006) 699–704.
[29] C. Carrascosa, J. Bajo, V. Julian, J.M. Corchado, V. Botti, Hybrid multi-agent architecture as a real-time problem-solving model, Expert Systems With Applications 34 (2007) 2–17.
[30] F. Fdez-Riverola, E.L. Iglesias, F. Díaz, J.R. Méndez, J.M. Corchado, Applying lazy learning algorithms to tackle concept drift in spam filtering, Expert Systems With Applications 33 (2007) 36–48.
[31] S. Chaudhury, T. Singh, P.S. Goswami, Distributed fuzzy case based reasoning, Applied Soft Computing Journal 4 (2004) 323–343.
[32] N.B. Karayiannis, G.W. Mi, Growing radial basis neural networks: merging supervised and unsupervised learning with network growth techniques, Neural Networks, IEEE Transactions 8 (1997) 1492–1506.
[33] N.R. Jennings, M. Wooldridge, Applying agent technology, Applied Artificial Intelligence 9 (1995) 351–361.
[34] M. Georgeff, A. Rao, Rational software agents: from theory to practice, in: N.R. Jennings, M.J. Wooldridge (Eds.), Agent Technology: Foundations, Applications, and Markets, Secaucus/Springer-Verlag, NJ/New York, 1998.
[35] E. Cerami, Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI & WSDL, O'Reilly & Associates, Inc., 2002.
[36] G.H. Dunteman, Principal Components Analysis, Newbury Park, California, 1989.
[37] R.W. Preisendorfer, Principal Component Analysis in Meteorology and Oceanography, Development in Atmospheric Science (1988).
[38] J.E. Fowler, Compressive-projection principal component analysis, IEEE Transactions on Image Processing 18 (2009) 2230–2242.
[39] P. Filzmoser, K. Hron, C. Reimann, Principal component analysis for compositional data with outliers, Environmetrics 20 (2009) 621–632.

[40] S.W. Lin, S.C. Chen, PSOLDA: a particle swarm optimization approach for enhancing classification accuracy rate of linear discriminant analysis, Applied Soft Computing 9 (2009) 1008–1015.
[41] V. Rokhlin, A. Szlam, M. Tygert, A randomized algorithm for principal component analysis, SIAM Journal on Matrix Analysis and Applications 31 (2009) 1100–1124.
[42] V. Ravi, C. Pramodh, Threshold accepting trained principal component neural network and feature subset selection: application to bankruptcy prediction in banks, Applied Soft Computing 8 (2008) 1539–1548.
[43] S. Gunter, N.N. Schraudolph, S.V.N. Vishwanathan, Fast iterative kernel principal component analysis, Journal of Machine Learning Research 8 (2007) 1893–1918.
[44] B. Fritzke, Growing cell structures – a self-organizing network for unsupervised and supervised learning, Neural Networks 7 (1994) 1441–1460.
[45] S. Haykin, Neural Networks, Prentice Hall, Upper Saddle River, NJ, 1999.
[46] F. Ros, M. Pintore, J.R. Chrétien, Automatic design of growing radial basis function neural networks based on neighboorhood concepts, Chemometrics and Intelligent Laboratory Systems 87 (2007) 231–240.
[47] E. Plaza, E. Armengol, S. Ontañón, The explanatory power of symbolic similarity in case-based reasoning, Artificial Intelligence Review 24 (2005) 145–161.
[48] H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, The Annals of Mathematical Statistics 18 (1947) 50–60.