12th International Conference on Information Fusion
Seattle, WA, USA, July 6-9, 2009

Solving disagreements in a Multi-Agent System performing Situation Assessment

Giuseppe Paolo Settembre, Daniele Nardi
DIS Dept., University "Sapienza" of Rome
lastname@dis.uniroma1.it

Alessandro Farinelli
DI Dept., University of Verona
alessandro.farinelli@univr.it

Roberta Pigliacampo
SESM s.c.a.r.l. - a Finmeccanica company
rpigliacampo@sesm.it

Mirco Rossi
SELEX S.I. - a Finmeccanica company
mirossi@selex-si.com

Abstract – *Situation Assessment and decision making in monitoring and surveillance scenarios are evolving from centralized models to high-level, reasoning oriented, net-centric models, according to new information fusion paradigms proposed by recent research.*

In this paper, we propose the design of a multi-agent architecture for Situation Assessment, where situations are classified through agent collaboration, in order to provide human operators with a synthetic vision, that points out the elements of the scenario that require human intervention. Specifically, our approach provides for: i) high level classification of situations based on OWL ontology reasoning; ii) distributed assessment by a protocol which solves disagreements possibly arising among agents' conclusions.

Experiments in a real maritime surveillance scenario show that the proposed approach approximates the results of a centralized architecture, while preserving independency of decision makers and dramatically reducing the amount of communication required.

Keywords: Situation Assessment, Multi-Agent Systems, Coordination, Description Logics.

1 INTRODUCTION

In modern applications, decision making processes often require to coordinate several actors. For example, in large-scale, complex domains like space explorations, disaster response, real-time monitoring, defense and security systems, collaboration is critical in order to achieve a prompt reaction to many unexpected situations that may happen. In particular, among the several challenges, a current objective, both from the scientific and industrial perspective, is to eliminate the dependency from a centralized decision entity, relying instead upon mechanisms which allow for decision-making in a distributed way.

In other words, Situation Assessment [2] is a process that "dynamically attempts to develop a description of current relationships among entities and events in the context of their environment". In other words, it aims at fusing information from different sources in order to recognize high-level relationships in a complex scenario, which deals with several actors and events.

Several techniques allow current systems to achieve effective low level analysis (level 0 and 1 of the JDL model [5]), thus we focus only on Situation Assessment. Very often, the classical approaches to high level information fusion are based on a centralized architecture [7, 4], or aim at refining perceptions at data level [1], instead of allowing for higher level conclusions. Among the multi-agent approaches to data fusion, some works based on probabilistic models have been proposed [6, 15, 13], but they are generally used for low level fusion (distributed sensing), while high level fusion needs, in general, more informative interpretation of data. In [14], a multi-agent system is used, but agents handle only feature level perceptions. A similar approach is the event based information fusion [8], where semantics of information is considered; however, the distributed information sources are able to perform only the feature extraction process, while fusion still relies on a single system entity.

In this paper, we present a multi-agent approach to Situation Assessment. Such solution is key whenever the application domain is endangered to become unmanageable with actual centralized solutions, due to the foreseeable increase of entities involved (e.g. maritime surveillance, air traffic control [11]). In order to achieve this goal, it is necessary to devise techniques which can convey pieces of information towards those destinations where they are strategic for decision making. *"The right information, in the right place, at the right time"*.

Our approach provides the following innovative contributions: (1) We formulate Situation Assessment as a multi-agent coordination process, where each agent's knowledge base (KB) and reasoning capability is represented in OWL-DL [10], a well known standard for high level reasoning; (2) we apply and adapt a coordination protocol for multi-agent situation assessment [13], in order to work for high level conclusions; (3) we present a validation of the approach in a real case maritime surveillance scenario.

About the first contribution, we formalize the problem using a set of cognitive agents that share a symbolic representation of the domain and, through collaboration, are able to classify situations of interest in order to share high level conclusions. The standard used for formal knowledge representation (ontology web language - OWL-DL [10]) is based on the formalism of Description Logics (DL) [9], which allow each agent for the classification of high level situations over a set of events, perceived with noise.

The second contribution of this paper is to address the core of the multi-agent Situation Assessment, which is to let the agents assess obtained conclusions, that may differ due to the different owned perceptions. We apply a distributed algorithm for situation assessment [13], which solves disagreements among agents perceived events, by using dialogues based on sequences of one to one interactions. While, in [13], only events were addressed, our extension is that here we let the dialogue be feasible in the context of a OWL-DL knowledge bases, allowing for a major expressivity and a standard representation for high level situations. In Sec.4.1, we present a viable solution, which requires to introduce in the agent's KB which events take part to the classification of each high level conclusion.

A validation in a real case context of maritime surveillance has been conducted in collaboration with Selex-SI, the system integrator house within the Finmeccanica group, which started addressing a net-centric architecture for harbour protection in the early 2000s, and is currently exploring the potentiality of multi-agent technologies in order to extend their system with more sophisticated decision making. Experiments in the maritime scenario will show that our approach requires a smaller amount of knowledge to be exchanged among agents than other solutions that broadcast each assertion, thus preserving locality and dramatically limiting communication requirements.

The paper is organized as follows. In Sec.2, we present our multi-agent approach to Situation Assessment; in Sec.3, we describe in detail the mapping from feature extraction to situation assertions in a OWL-DL KB; in Sec.4, the issues specifically concerning the multi-agent architecture are addressed. In Sec.5, we present the set of experiments performed in the context of maritime surveillance.

2 A multi-agent approach to Situation Assessment

In Fig.1, the main steps of a multi-agent approach to Situation Assessment are shown. Squares represent the agents that take part to the process. Fires, emoticons and lightnings represent dynamic events which are imperfectly perceived by agents.

In the first step of the process, denoted as "Perception and feature extraction", distributed sensing is performed, where each agent separately performs perception with his own sensors, and extracts some relevant features from sensor readings. This part of the process includes also data association (or object assessment).

The second part of the process, denoted as "Situation Classification", consists in obtaining, from a single agent perspective, a high-level assessment hypothesis of situations which are of interest in the perceived scenario. Each agent may estimate that several situations are currently present in the scenario[1]. Due to different agent's perceptions, agents may have different assessment at the end of this phase.

The third part of the process, denoted as "Multi-agent assessment", consists in the agents arguing, generally through message exchange, about their assessments, aiming at reaching an agreement on each of their previous conclusions. At the end of this phase, the team as a whole has somehow solved the conflicting assessments. In general, it is not necessary that each agent knows the assessment of each situation, because it may not be interested in having information about certain situations at all.

The last part of the process, denoted as "Task assignment and execution", involves planning the best intervention, assigning tasks to the agents and execute them. Some parts of this phase may again be performed in a distributed way.

Our research will focus precisely on the two central parts of the process. The notation and the definitions which are presented in the remaining of the paper will deliberately focus only on them, considering the remaining ones out of the scope of this research.

2.1 Problem formalization

We considered a certain number of observed entities pursuing some unknown private goals, and a set of agents $A_1, ..., A_m$. Each agent has a world model, its own perceptions, it communicates with the other agents, and takes part pro-actively in the classification process. The agent's world model is an *ontology*, which is a symbolic representation of the organization of concepts and their relevant relationships into the domain (intensional knowledge, TBox), plus a set of *assertions* on individuals (extensional knowledge, ABox); the TBox is shared among all the agents, and constitutes a common language for communication.

An event e_i is represented as a logical condition, which denotes a specific feature of interest, and its detection is based on the observations (perceptions) on the environment.

In order to extract only few relevant elements of the perceived events, we consider the set of the *relevant situation classes* $S_{CL} = \{S_1, \ldots, S_n\} \cup S_\top$, in which each S_i identifies a type of situations, of interest with respect to the context. Basically, each relevant situation class S_i represents a group of semantically equivalent circumstances of the world (for the purpose of Situation Assessment). A symbolic definition of each S_i is given in terms of the type of events which are observable in the environment. In the following, the situation classes are defined in Description Logics (DL) [9]. With $S_\top$ we denote the most generic situation class, which

[1]The presence of several situations in the agents' balloon may be misleading the reader who is familiar with the term *situation* in formal logics, because having several situations would represent that an agent has several alternative models for the KB. This is *not* the intended meaning: speaking in terms of formal logics, we may say that we indicate with different terms independent conclusions extracted from the KB typically corresponding to a single model (the global situation).

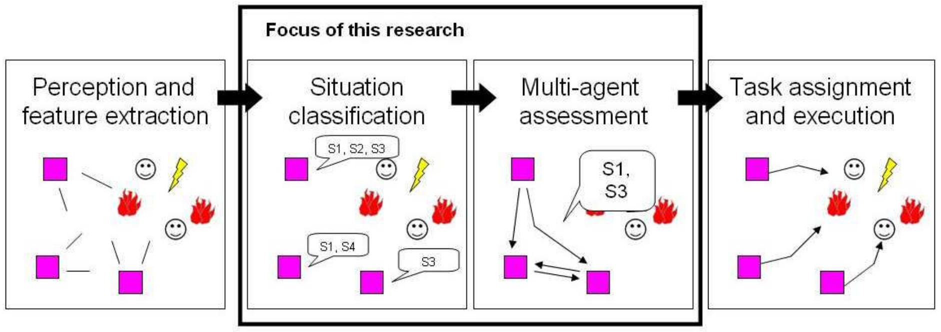

Figure 1: Description of a multi-agent situation assessment process.

includes all possible situations. A situation class S_i is *more specific* than S_j, iff every instance of S_i is also an instance of S_j. In DL, this is indicated by $S_i \sqsubseteq S_j$.

A situation instance s_i is an individual characterized by a set of events, which are actually observed by an agent on the scenario. From the point of view of the agent's knowledge, with $K_p(s_i \in S_j)$ we denote that an agent A_p *knows that* $s_i \in S_j$, which means that a set of events, which characterize s_i as in the definition of a class S_j, are known to A_p: in DL, this simply means that in A_p's knowledge base, s_i is classified as an instance of S_j.

We can now provide a formal definition of the Situation Assessment process.

Definition 1 We say that s_i *is assessed as* S_i by an agent A_p iff

- $K_p(s_i \in S_i)$
- $\nexists S_j\ s.t.\ K_p(s_i \in S_j) \wedge S_j \sqsubseteq S_i$.

Thus, given the above formalization, *Situation Assessment (SA)* is the *non-trivial* assessment of all the instances s_i. A classification of s_i as S_i is *non-trivial* if $S_i \neq S_\top$.

3 Situation classification: event assessment

In order to populate the agent's ontology with data extracted from the agent's information sources, we have to explicitly deal with the uncertainty of event perception. The connection between agent's KB and lower level fusion does not impact on our approach, thus we chose the simplest possible solution, a Bayesian filter: devising a more sophisticated technique on this respect is out of the paper's scope.

Each time a new sensor reading is provided, features are extracted about specific circumstances of interest, and they are abstracted as positive or negative observations of certain events at specific time instances. Algorithm 1 presents the pseudo-code executed by an agent, when a new observation is received (directly from sensors or another agent). A simple Bayesian filter is instantiated to obtain a belief about each detected event in the environment, which considers all the observations in the past, and the ones received from other agents. When a reading referring to a past time is obtained, the filter is reinitialized, and the belief recalculated, starting from the time of the oldest observation. Once a belief is available, a simple rule allows an agent to add assertions to the agent's ABox (the facts in the KB).

Algorithm 1 INTEGRATEOBSERVATION

```
Input: an observation $obs$
Output: update of agent's list of observations and ABox

1: $event\_Obs \leftarrow$ GETOBS$(event)$
2: **if** $obs \notin event\_Obs$ **then**
3:     $event\_Obs \leftarrow event\_Obs \cup obs$
4:     $belief \leftarrow$ EVOLVE_FILTER$(event\_Obs)$
5:     REVISE( $event$ )
6:     REVISE( $\neg event$ )
7:     **if** $belief > k_1$ **then**
8:         ASSERT( $event$ )
9:     **if** $belief < 1 - k_1$ **then**
10:        ASSERT( $\neg event$ )
```

Algorithm 1 allows each agent to know whether considering an event to be true, false or unknown, given his available observations. Thus, in our approach, *each ABox assertion is "justified" by a list of observations, which can be retrieved at any time, if needed.* The function GETOBS retrieves from the agent's memory the list of observations related to a certain assertion. The functions ASSERT/REVISE are used to add/remove assertions from the agent's assertion ontology (ABox). REVISE has no effect if the assertion is not already present in the KB. Notice that, when using an open world logic -like DL are-, $ASSERT(\neg e)$ has a different meaning than $REVISE(e)$ because the first one asserts the fact e to be false, the second one leaves it unknown.

For the sake of simplicity, in lines 7-9, the condition to assert an event (or its negation) into the KB has a single parameter k_1, which expresses the degree of certainty required to believe it true or false, however other more sophisticated techniques may be used.

3.1 Situation classification

The taxonomy, which is shared by all agents, has been illustrated in [12]. Each agent is provided with a taxonomy of the relevant events of the domain, and one which includes the definitions of the relevant situation classes.

In the following sections, we will build on an example placed in the domain of maritime surveillance. We define the following concept names: C_F is a free access sea area, C_L is a lockout sea area, C_N is a not navigable area, C_D is a dangerous area, C_U is an unidentified vessel, C_W is a vessel with weapons onboard. Moreover, R_1 is the property which relates a vessel with its position. We will consider that all agents are provided with the following TBox assertions:
(1) Free access zones are disjoint from lockout zones
(2) Lockout zones are the union of not navigable and dangerous areas
(3) A vessel has a unique position
(4) S_1 is the set of unidentified vessels
(5) S_2 is the set of vessels detected in a lockout zone
(6) S_3 is the set of vessels both unidentified and armed
and agent A_1 is provided with ABox:
(7) a_1 is a position in a free access zone
(8) a vehicle s is detected in position a_1
(9) vehicle s is unidentified
(10) vehicle t is unidentified

Assertions (1),(2),(3) refer to a simple taxonomy of the scenario, while (4),(5),(6) are a simple taxonomy for situation classes. In Description Logics, the above sentences are written as follows:
(1) $C_F \equiv \neg C_L$
(2) $C_L \equiv C_N \sqcup C_D$
(3) $\top \sqsubseteq \leq 1\, R_1$
(4) $S_1 \equiv C_U$
(5) $S_2 \equiv \exists R_1.C_L$
(6) $S_3 \equiv C_U \sqcap C_W$
and agent A1's ABox is:
(7) $C_F(a_1)$
(8) $R_1(s, a_1)$
(9) $C_U(s)$
(10) $C_U(t)$

The situation instances are classified and assessed using standard DL inference capabilities. The classification process is also explained in details in [12]. The specific reasoning service used is implemented in common Description Logics reasoners, and it is called *Realization* [9], whose results meet, by definition, the requirements of definition 1. In the following, we will call EVALASSESSMENT on a specific instance s the result of a *Realization* query over s. Thus, we do not need to use any rule propagation engine, unlike [7].

The approach that we have just presented to situation management may still lead to inconsistent information be introduced into the agent's KB. Notice that, since the intensional knowledge (TBox) is shared among the agents, we can reasonably assume it is consistent and contains all the relevant cases which can actually happen in the perceived environment. Thus, we only have to deal with inconsistent assertions about individuals; a generic inconsistency manager can be used to retrieve the subset of assertions which create the inconsistency.

4 Multi-agent assessment

A coordination technique is necessary, in order to agree on a single classification, since agents may reach different conclusions, due to their partial (and noisy) perceptions. We have chosen to extend the coordination protocol proposed in [13], that we briefly describe here.

When an agent is able to formulate locally a non-trivial situation assessment proposal, it sends a proposal for that conclusion. The proposal is sent through the agents that, when receiving the assessment proposal, evaluate it (EVALARGUMENTS). In case they disagree, instead of forwarding the message on, they challenge the proposal, sending back the observations that are conflicting (RETRIEVEREFUTINGARGUMENTS). The agent receiving the challenge integrates these observations into its own beliefs (as in Sec.3) and reconsiders a change in the classification. In case the additional observations do not cause a change in the assessment, the agent sends to the challenger the list of observations (RETRIEVESUPPORTINGARGUMENTS) that are needed solve the conflict; vice-versa, if the agent changes its assessment, it sends the proposal back to where it received it, attaching relevant observations and asking again for agreement; the challenge continues until it is solved, or the proposal destroyed. Once a sufficient number of agents agree with the proposal, the assessment is completed.

Every disagreement is solved by the algorithm in [13] through exchange of observations, using an ad-hoc KB to allow for simple retrieval of justifications. If agents are provided with a KB, where conclusions are obtained through logic inference, a more general solution must be addressed.

4.1 Retrieving justifications

The algorithm in [13] may be used with a generic knowledge base, if three key functions can be provided:

- EVALARGUMENTS.
  Input: an assessment proposal P on an instance s.
  Output: a boolean value.
  This function returns true if the executing agent agrees with a received proposal P on the situation instance s, false otherwise.

- RETRIEVEREFUTINGARGUMENTS.
  Input: an assessment proposal P on an instance s.
  Output: a set of arguments.
  If the executing agent disagrees with P, it returns the arguments to attack P.

- RETRIEVESUPPORTINGARGUMENTS
  Input: an instance s, a set of arguments.
  Output: a set of arguments.

The executing agent, who receives a set of challenging arguments, evaluates the arguments to attack the challenge, and the ones to support its proposal.

We define arguments as $\langle C(i_1), obsList\rangle$ or $\langle R(i_1, i_2), obsList\rangle$, where C is a concept name, R is a role name, i_1 and i_2 are instance names, *obsList* is a list of sensor feature observations. The intuitive meaning of an argument $\langle C(i_1), obsList\rangle$ ($\langle R(i_1, i_2), obsList\rangle$) is that the observations related to the fact that $C(i)$ ($R(i_i, i_2)$) are valid assertions into an agent's KB, and they are supported by the observations in *obsList*. Instead, if $obsList = \emptyset$, the argument indicates that the fact is unknown to the agent.

We assume, as it is in the example, that the agent's KB is consistent before starting the agent dialogue. (If it is present, an inconsistency in the agent's KB must be handled first).

EVALARGUMENTS When receiving an assessment proposal S_i for a situation instance s, evaluating the agreement means verifying whether the *best assessment* of s is S_i. Recalling Definition (1), the Algorithm EVALARGUMENTS will yield *true* if EVALASSESSMENT on s yields S_i, *false* otherwise.

RETRIEVEREFUTINGARGUMENTS This function is executed whenever agent A_1 receives an assessment proposal and he does not agree with it. In our example, agent A_1 will disagree, if he receives a proposal claiming $S_2(s)$; similarly, if the proposal claims $S_3(s)$. In both cases, he must compute the set of arguments to attach in a reply message to challenge the proposal.

Let's analyze what the correct answer would be in our example. In case the agent receives $S_3(s)$, the agent disagrees because he lacks of certainty about $C_W(s)$. In this case, the agent's answer will include the argument $\langle C_W(s), \emptyset\rangle$, which means "To the best of my ($A1$'s) knowledge, it is unknown s being of class C_W".

Let's analyze now the case of the agent receiving $S_2(s)$. In this case, he disagrees because:
i) Instance s participates to R_1 with instance a_1. Since, (3) claims that R_1 is functional, there can not be another instance, related to s through R_1;
ii) the agent is not aware of a_1 being instance of C_N or C_D, but it does for C_F that, according to (1), is disjoint from C_L;
iii) The agent knows that his best assessment of s is S_1, because he knows s being instance of C_U.
Notice that, while the agent answers i) and ii) concern challenging the other agent's proposal, iii) aims at justifying his own assessment. Moreover, the agent did not justify why S_3 is not his best classification, because that would be off topic. Excluding TBox assertions, the agent should then challenge with observations related to assertions (7), (8), (9), and claiming unknown arguments C_N and C_D.

The mechanism we use to retrieve the justifications currently requires the ontology designer to specify, for each situation class, the subset of arguments that may allow for the classification of an instance to that given class. In particular, she is in charge of providing the sets $JUST_{S_i}(TBox, T)$, which include, for each S_i the set of all possible template assertions that would allow the agent to classify a *generic* instance T to class S_i. These sets are dependent only on the TBox, which is shared among the agents. E.g. in the TBox of the example:
$JUST_{S_1}(TBox, T) = \{C_U(T)\}$
$JUST_{S_2}(TBox, T) = \{R_1(T, T_2), C_N(T_2), C_D(T_2), C_L(T_2)\}$
$JUST_{S_3}(TBox, T) = \{C_U(T), C_W(T)\}$

Assuming that agents are provided with these sets, it is easy to define a function $JUST(KB, S_i, s)$ which retrieves, using a predefined procedure, from the knowledge base KB a set of possible assertions, concerning a *specific* instance s, which are related to the situation class S_i. In particular, the function $JUST$ retrieves the set $JUST_{S_i}$, and applies it on the particular agent's ABox at execution time: the implementation of the function $JUST$ is general, and consists in the matching of the template assertions with actual instances. The results of JUST, applied on the instances of the above ABox, for each situation class are:
$JUST(KB, S_1, s) = \{C_U(s)\}$
$JUST(KB, S_2, s) = \{R_1(s, a_1), C_N(a_1), C_D(a_1), C_L(a_1)\}$
$JUST(KB, S_3, s) = \{C_U(s), C_W(s)\}$
$JUST(KB, S_1, t) = \{C_U(t)\}$
$JUST(KB, S_2, t) = \{\}$
$JUST(KB, S_3, t) = \{C_U(t), C_W(t)\}$

Finally, notice that $JUST$ does not provide any reasoning service, and, correctly, it may return also assertions which are not in the ABox (in the example above, $C_N(a_1), C_D(a_1), C_L(a_1), C_W(s), C_W(t)$ are not in the agent's KB): these will let the agent retrieve also the unknown assertions to attach.

Assuming agent is provided with the function $JUST$, the pseudo-code for RETRIEVEREFUTINGARGUMENTS is shown as Algorithm 2.

Algorithm 2 RETRIEVEREFUTINGARGUMENTS

```
Input: a classification $C(o_1)$
Output: a set of arguments

1: $RET\_SET \leftarrow \emptyset$ //set of refuting arguments
2: $ASSERT\_SET \leftarrow \emptyset$ //set of refuting assertions
3: $J_1 \leftarrow JUST(KB, C, o_1)$
4: for all $s \in J_1$ do
5:     if $TBox \cup ABox \nvDash s$ then
6:         $ASSERT\_SET \leftarrow ASSERT\_SET \cup s$
7: best $\leftarrow$ EVALASSESSMENT($o_1$)
8: $J_2 \leftarrow JUST(KB, best, o_1)$
9: for all $s \in J_2$ do
10:     $ASSERT\_SET \leftarrow ASSERT\_SET \cup s$
11: for all $a \in ASSERT\_SET$ do
12:     $RET\_SET \leftarrow RET\_SET \cup \langle a$,GET_OBS$(a)\rangle$
13: return $RET\_SET$
```

We provide an example, following again the case where the agent receives a proposal for $S_2(s)$. First (lines 3-6), some arguments to challenge will be extracted from

$J_1 = JUST(KB, S_2, s) = \{(7), (8), C_N(a_1), C_D(a_1)\}$. The observations related to $\{(7), (8)\}$ will be attached to the message. Moreover, the arguments $\langle C_N(a_1), \emptyset\rangle, \langle C_D(a_1), \emptyset\rangle$ will be attached. Finally (lines 7-10), agent A_1 will justify its best assessment S_1 and will add also observations related to the KB assertion (9) to the message.

Now, consider the case where the agent receives a proposal for $S_3(s)$. Again, $KB \nvDash S_3(s)$, therefore A_1 does not agree. $J1 = \{C_U(s), C_W(s)\}$. Then, A_1 will preliminarily attach only the fact that $C_W(s)$ is unknown, since he agrees on $C_U(s)$. Finally, to justify its own best assessment, he will retrieve $J_2 = \{C_U(s)\}$, and attach also $C_U(s)$. This is a small lack of efficiency, since, in this case, A_1 is sure already that the agent who has proposed S_1 knows the argument related to $C_U(s)$; even if it would be easy to solve the cause of this inefficiency in this example, it would be difficult to detect it in the general case.

RETRIEVESUPPORTINGARGUMENTS This function is executed by an agent, after he received a reply to a proposal with a set of arguments attached, and he integrated the observations in the arguments, and evaluated that he did not change his best assessment. The aim of the function is to estimate which are the arguments to attach to a reply back to the challenger to guarantee his agreement.

We use a similar policy for this function, as for RETRIEVEREFUTINGARGUMENTS. We use again the function $JUST$, that has been defined in the previous paragraph. Algorithm 3 for retrieving supporting arguments slightly differs from the one for computing refuting arguments. In particular, the agent will have to consider the arguments that have been sent to him, and attach its related observations on each of the arguments. Moreover, it will have to attach the justifications for his best assessment.

Algorithm 3 RETRIEVESUPPORTINGARGUMENTS

```
Input: a situation instance $o_1$, a set of received arguments $ARG\_RCVD$
Output: a set of arguments

1:  $RET\_SET \leftarrow \emptyset$ //set of support arguments
2:  $ASSERT\_SET \leftarrow \emptyset$ //set of support assertions
3:  for all $\langle a, obsList\rangle \in ARG\_RCVD$ do
4:      $ASSERT\_SET \leftarrow ASSERT\_SET \cup a$
5:  $best \leftarrow$ EVALASSESSMENT($o_1$)
6:  $J_1 \leftarrow JUST(KB, best, o_1)$
7:  for all $s \in J_1$ do
8:      $ASSERT\_SET \leftarrow ASSERT\_SET \cup s$
9:  for all $a \in ASSERT\_SET$ do
10:     $RET\_SET \leftarrow RET\_SET \cup \langle a,$GET_OBS$(a)\rangle$
11: return $RET\_SET$
```

In the above described approach, the ontology designer is in charge of defining the sets $JUST_{S_i}$, for each situation class S_i. Sometimes, she may exploit this property of the system to deploy different sets of possible template justifications for each situation class, and verify the performance of the system, when attaching only a small subset of events. For example, she may allow for several iterations of a dialogue between two agents, by sending most probable justifications first, then, only if the disagreement is not solved, the rare ones. However, in most cases, it will be probable that the designer will not want to define these sets, and just be sure that the system will attach all possible relevant information to solve the challenge and avoid multiple iterations. Results from the *belief revision* theory [3] may be used, where it is evaluated the minimal amount of assertions which cause an inconsistency inside a KB. Even if the justifications are evaluated through logic reasoning, the experiments shown in Sec.5 should not be affected, because the results of the function $JUST$ is identical in the two implementations.

5 Experiments and results in a sea-coast surveillance scenario

This evaluation has been performed on a multi-agent platform for maritime surveillance. Agents represent patrol-ships and command and control workstations. This multi-agent platform is a component of a general architecture, from which it receives in input perceptions in terms of objects with an uncertain estimated position and a unique (possible incorrect) ID; other inputs are available through database access. On the output side, the software architecture provides a graphical interface, which shows every assessed situation as a warning on a display, to allow for human decision making. The multi-agent architecture can be run on a laptop with single core processor, with small sized teams of cognitive agents (from 1 up to 5 members) and about 100 external entities moving. The number of agents is limited due to the high amount of radar data received and of real time reasoning service requests over the ontologies.

With respect to the quality of the input data, the noise in the radar data can not be eliminated, therefore it will be always present. Thus, in our experiments, we will consider three different settings. We call "high quality perception" the setting where we consider only sensors' noise. Then, we reduce the quality of the feature extraction process through an exponential decay factor, which increases observations' noise in relation to the distance from the perceived event: "medium quality perceptions" refers to exponential decay equal to 0.01, and "low quality perceptions" equal to 0.1.

We focus on various suspect operations to be detected. Among the others, for example:
splitting: it is the manoeuver of remaining hidden staying close to another vessel, then suddenly move away directed to a critical area. (see Fig.2).
suspect approach: it is the case of a suspect vessel approached by other (at least two) vessels. A suspect vessel is a vessel whose identification is not known, which stays near the border of a surveilled zone.

Other situation classes are present. For example, there are some partially specified situations, which are general-

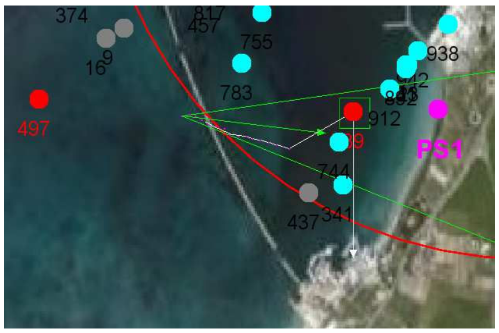

Figure 2: A splitting situation. Two boats (red circles) performed a split close to a surveillance area (red arc) and one of them is directed to the critical point (in purple).

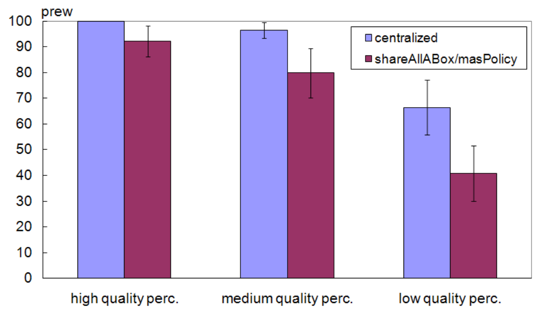

Figure 3: Performances varying quality of perceptions.

izations of the above two. While detecting this high level situation, agents may have a disagreement on different assertions, like the region were the vessel first appeared, the average direction of the vessel (which is obtained reasoning also on past perceptions) or the number of vehicles involved in the procedure.

Our approach (referred as *mas_Policy*) was compared to two different strategies. The first one, *centralized*, represents an approach where the high level fusion process is performed by a specific agent (which represents a command and control center) and patrolships which are distributed in the sea area considered, are used only as information sources. The second benchmark policy, *share_all_ABox*, represents the multi-agent solution, where agents execute the proposed algorithm to reach agreements, solving their disagreements by attaching all assertions in the ABox.

Preliminarily, we show the quality of the overall team assessment, comparing our policies, under varying noise conditions. In this case, the *mas_Policy* and *share_All_ABox* are undistinguishable, because all the relevant observations are shared, in both policies. On the x axis, the perception quality is shown. On the y-axis there is an indicator of performances *prew* (% of reward). *prew* uses an utility function which gives rewards and costs to correct/wrong assessments and partially correct/wrong ones. Then, $prew = \frac{u^*}{u_{max}}$, where u^* is the utility that the team achieves through its classifications, and u_{max} is the maximum achievable utility, considering the set of classifications of the *centralized* policy, in the "high quality" configuration, as ground truth (correct assessment).

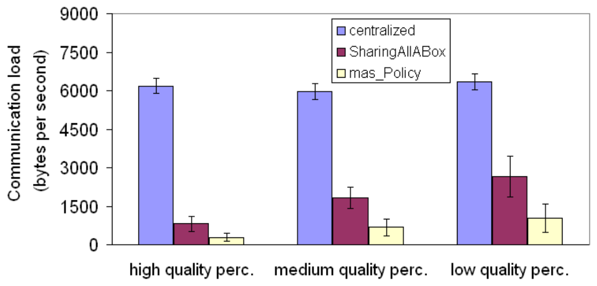

Figure 4: Communication costs of 3 different agent policies.

The graph in Fig.3 highlights three aspects. The first one is that also the *centralized* approach does not get always the best result, because it suffers as well from the quality of received observations. Secondly, the centralized approach always over-performs the multi-agent policy, as expected. This happens because the *centralized* approach has always instantaneous access to all available observations. Finally, in case of low level perceptions the quality of results of the multi-agent policy is sensibly affected: in this case, the approach suffers from the fact that a small number of agents is used, and loosing some observations may result key to a correct classification. Increasing the density of agents would improve performances, as it is shown in the evaluation of the chosen algorithm for coordination [13]; also, exploiting the locality of information to forward proposals to the best informed agents would enhance the performances with respect to the actual random forwarding policy. In addition, we evaluate communication costs, in the three different policies. We analyze the amount of bandwidth used, which is a better indicator of the costs than the number of messages, because messages have a different size in the various policies, due to the possible presence (and the number) of observations attached. We considered a fix size of 100 Bytes for each observation. In Fig.4 we show the results. On the x axis, the quality of observations is shown, while the y axis indicates the bandwidth used, measured in bytes per second. We can observe that *centralized* has severe bandwidth requirements, while the two multi-agent approaches significantly reduce the amount of bandwidth used. The used bandwidth of the approaches with distributed fusion is slightly influenced by the quality of perceptions, since the number of conflicts in the assessment increases. Even for the worst case, with very noisy observations, the policies with distributed fusion largely reduce (more than half) the amount of necessary bandwidth. Moreover, comparing the *share_all_ABox* and the *mas_Policy* approaches, the second one further reduces (about one order of magnitude less than *centralized*) the bandwidth used, because the minimal amount of events to be challenged is evaluated.

A major reason to compute and share minimal informa-

	high quality perc.		medium quality perc.		low quality perc.	
	assert.	assert.shared	assert.	assert.shared	assert.	assert.shared
centralized	177	177 (100%)	673	673 (100%)	596	596 (100%)
shareAllABox	110	110 (100%)	151	151 (100%)	423	423 (100%)
mas_Policy	110	12 (11%)	148	31 (21%)	439	123 (28%)

Table 1: Measuring information locality. The number of assertions and of shared assertions of each agent is shown.

tion among agents is to preserve information locality. To evaluate this attribute, in Table 1 we related it to the amount of ABox assertions which are shared among the agents. In particular, we use the ratio between the number of assertions shared and the total amount of assertions in the KB (both averaged among the agents). We count as ABox assertions, only those that derive from observations (and not, for example, static facts). Moreover, changes in the value (positive, negative, or unknown) of an assertion are correctly counted just once. We consider an assertion to be shared, whenever an observation related to that assertion is sent. Results are shown in the table, where we may conclude that, with our policy, each agent shares less than 30% of its KB. The meaning of assertions that are not shared indicates that either they are useless for current conclusions, or that they are agreed (perhaps for different reasons) among the agents. We see also that the $share\_all\_ABox$ and the $centralized$ policy are worse with respect to locality, since they share all the local assertions.

6 Conclusions

This paper is an important step towards a multi-agent approach to Situation Assessment. It provides a general framework, and a solution encompassing all the components of the process. In particular, it uses the standard logic inference for situation analysis, and addresses the comparison of different conclusions, by using dialogues between agents.

The experimentation in a real case scenario of maritime surveillance has shown that the approach dramatically reduces the amount of necessary communications. Moreover, it allows to obtain meaningful high level assessments, merging conclusions of several agents, while preserving the specificity of each agent.

Several aspects of the overall process could be improved. We are concerned specifically on the agent interaction leading to high level assessment. For example, a future work in this area will focus on the possibility to compute the set of justifications for each relevant situation class, using results from belief revision theory [3]. Another possible direction to investigate is whether an agent may solve also inconsistencies (and not only disagreements) in its own KB by asking for an agreement to team mates, attaching its subset of inconsistent assertions.

References

[1] G. T. Capraro, A. Farina, H. Griffiths, and M. C. Wicks. Knowledge-based radar signal and data processing. *IEEE Signal Processing Magazine*, 23:18–29, Jan. 2006.

[2] D. L. Hall and J. Llinas. *Handbook of Multisensor Data Fusion*. CRC Press, 2001.

[3] A. Kalyanpur. *Debugging and Repair of OWL Ontologies*. PhD thesis, Univ. of Maryland, College Park, USA, 2006.

[4] E. G. Little and G. L. Rogova. Designing ontologies for higher level fusion. *Inf. Fusion*, 10(1):70–82, Jan. 2009.

[5] J. Llinas, C. L. Bowman, G. Rogova, A. N. Steinberg, et al. Revisiting the JDL data fusion model II. In *Proc. of the 7th Int. Conf. on Information Fusion (IF-04)*, pages 1218–1230, Stockholm, Sweden, June 2004. ISIF.

[6] A. Makarenko and H. F. Durrant-Whyte. Decentralized data fusion and control algorithms in active sensor networks. In *Proc. of the 7th Int. Conf. on Information Fusion (IF-04)*, pages 479–486, Stockholm, Sweden, June 2004. ISIF.

[7] C. J. Matheus, M. M. Kokar, and K. Baclawski. A core ontology for situation awareness. In *Proc. of the 6th Int. Conf. on Information Fusion (IF-03)*, pages 545–552, Cairns, Australia, July 2003. ISIF.

[8] N. Museux, J. Mattioli, C. Laudy, and H. Soubaras. Complex event processing approach for strategic intelligence. In *Proc. of the 9th Int. Conf. on Information Fusion (IF-06)*, pages 1–8, Florence, Italy, July 2006. IEEE.

[9] D. Nardi and R. J. Brachman. An introduction to description logics. In *The Description Logic Handbook*, pages 1–40. Cambridge University Press, 2003.

[10] OWL. www.w3.org/TR/owl-features/.

[11] Air transport framework - the performance target. Technical report, SESAR Consortium, 2006.

[12] G. P. Settembre, R. Pigliacampo, and D. Nardi. Agent approach to situation assessment. In Filipe, Fred, and Sharp, editors, *Proceedings of the 1st International Conference on Agents and Artifical Intelligence (ICAART-09)*, pages 287–290, Porto, Portugal, January 2009. INSTICC Press.

[13] G. P. Settembre, P. Scerri, A. Farinelli, K. Sycara, and D. Nardi. A decentralized approach to cooperative situation assessment in multi-robot systems. In *Proc. of the 7th Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, pages 31–38, Estoril, Portugal, May 2008.

[14] K. Sycara, R. Glinton, B. Yu, J. A. Giampapa, S. R. Owens, et al. An integrated approach to high level information fusion. *Information Fusion*, 10(1):25–50, Jan. 2009.

[15] B. Yu, P. Scerri, K. Sycara, Y. Xu, and M. Lewis. Scalable and reliable data delivery in mobile ad hoc sensor networks. In *Proc. of the 5th Int. Conf. on Autonomous Agents and Multi Agent Systems (AAMAS-06)*, pages 1071–1078, Hakodate, Japan, May 2006. ACM.