Heras Barberá, SM.; Jordan Prunera, JM.; Botti V.; Julian Inglada, VJ. (2013). Case-Based strategies for argumentation dialogues in agent societies. Information Sciences. 223:1-30. doi:10.1016/j.ins.2012.10.007.

Case-based Strategies for Argumentation Dialogues in Agent Societies

Stella Heras, Jaume Jordán, Vicente Botti, Vicente Julián

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València, Spain; Camino de Vera sn 46022 Valencia (Spain)

Abstract

In multi-agent systems, agents perform complex tasks that require different levels of intelligence and give rise to interactions among them. From these interactions, conflicts of opinion can arise, especially when these systems become open, with heterogeneous agents dynamically entering or leaving the system. Therefore, agents willing to participate in this type of system will be required to include extra capabilities to explicitly represent and generate *agreements* on top of the simpler ability to interact. Furthermore, agents in multi-agent systems can form societies, which impose social dependencies on them. These dependencies have a decisive influence in the way agents interact and reach agreements. Argumentation provides a natural means of dealing with conflicts of interest and opinion. Agents can reach agreements by engaging in argumentation dialogues with their opponents in a discussion. In addition, agents can take advantage of previous argumentation experiences to follow dialogue strategies and persuade other agents to accept their opinions. Our insight is that Case-Based Reasoning can be very useful to manage argumentation in open multi-agent systems and devise dialogue strategies based on previous argumentation experiences. To demonstrate the foundations of this suggestion, this paper presents the work that we have done to develop case-based dialogue strategies in agent societies. Thus, we propose a case-based argumentation framework for agent societies and define heuristic dialogue strategies based on it. The framework has been implemented and evaluated in a real customer support application.

Keywords: Argumentation, Agreement Technologies, Multi-Agent Systems, Case-Based Reasoning

1. Introduction

Nowadays, most large scale computer systems are viewed as service-oriented systems whose software components perform business transactions by providing and consuming services. These components are often implemented as agents in a Multi-Agent System (MAS). Agents are able to reason autonomously, represent human interests and preferences and group together to form societies that link them via dependency relations. In these societies, agents perform complex tasks that require different levels of intelligence and give rise to interactions among them. From these interactions, conflicts of opinion can arise, especially when MAS become open (with heterogeneous agents dynamically entering or leaving the system). Therefore, software agents willing to participate in this type of system will be required to include extra capabilities to explicitly represent and generate *agreements* [45].

Argumentation provides a natural means of dealing with conflicts of interest and opinion. Agents can reach agreements by engaging in argumentation dialogues with their opponents in a

Email address: sheras@dsic.upv.es (Stella Heras, Jaume Jordán, Vicente Botti, Vicente Julián)

discussion. Therefore, the study of argumentation in MAS has gained a growing interest [38, Part III]. Case-Based Reasoning (CBR) [1] is another research area where argumentation theory has produced a wide history of successful applications [46, 20]. According to the CBR methodology, a new problem can be solved by searching in a case-base for similar precedents and adapting their solutions to fit the current problem. This reasoning methodology has a high resemblance with the way by which people argue about their positions (i.e. beliefs, action proposals, opinions, etc.), trying to justify them on the basis of past experiences. Case-based reasoning systems have been recently applied to many different areas, such as help-desk environments [22], prediction markets [30], forecasting systems [10] or intrusion detection [33]. Also, the work done in the eighties and nineties about legal CBR fostered the argumentation research in the AI community [40]. From then on, the good results of CBR systems in argumentation domains suggest that this type of reasoning is suitable to manage argumentation processes [20].

On top of the simpler capacity to argue and reach agreements, agents can take advantage of previous argumentation experiences to follow dialogue strategies and persuade other agents to accept their opinions. Dialogue strategies allow agents to select the best argument to defend their positions in each step of the dialogue. Therefore, by following a dialogue strategy, an agent can achieve a more advantageous outcome from the dialogue or be able to improve the efficiency of the dialogue itself (shortening the agreement process, for instance). However, the social dependencies of agents and their preferences over values (the goods that agents want to promote or demote with their arguments) determine the way in which they can argue with other agents. For instance, agents that represent workers in a company are not as willing to accept arguments from equals as they are from superiors (due to a power dependency relation between superiors and other workers). Similarly, an agent representing a worker that acts as the trade union representative does not argue (for instance, to increase salaries and promote the value 'workers welfare') in the same way if it is defending the values of all workers as if it would be defending its individual values (for instance, to promote its 'reputation' within the company). Thus, we endorse the view of value-based argumentation frameworks [12, 11] which stress the importance of the audience (other participants in the dialogue) in determining whether an argument is persuasive or not. Therefore, we also consider values as an important element of the social context of agents.

Starting from the idea that the social context of agents determines the way in which agents can argue and reach agreements, this context should have a decisive influence in the computational representation of arguments, in the argument management process and in the way agents develop strategies to argue with other agents. While work on argument generation and evaluation has received much attention, the strategic use of arguments historically received little attention in the literature, although in the last five years this area has experienced a growing interest. Our point of view is that CBR can be very useful to manage argumentation in open MAS and devise dialogue strategies based on previous argumentation experiences. To demonstrate the foundations of this suggestion, this paper presents our approach to develop case-based dialogue strategies in agent societies. These strategies make use of a case-based argumentation framework that allows agents to argue and reach agreements taking into account their social context.

First, Section 2 briefly presents the case-based argumentation framework that we have developed to represent and manage arguments in agent societies. The framework takes into account the agents' social context in the way agents can argue. Thus, social dependencies between agents and their values are considered. Also, agents implementing this framework will be able to engage in argumentation dialogues following different dialogue strategies. With these strategies, agents will be able to select the most appropriate argument to bring about their desired outcome of the dialogue. Then, Section 3 shows the case-based dialogue strategies proposed in this work.

In Section 4 the framework is implemented in a real customer support application where agents represent technicians of a call centre that engage in argumentation dialogues to decide the best solution to apply for a problem reported to the centre. Therefore, each agent proposes its own solution and tries to justify it as the best solution to apply, persuading the other agents to accept its opinion. In this section different dialogue strategies that agents can follow are evaluated. Section 5 analyses related work and compares our approach with it. Finally, Section 6 summarises the main contributions of this work.

2. Case-based Argumentation Framework

In this section, we briefly introduce the argumentation framework that allow us to develop the case-based dialogue strategies presented in this paper (a detailed explanation can be found in [19, Chapter 3]). This framework has been implemented as an argumentation API in the Magentix2 agent platform and is publicly available at `http://www.gti-ia.upv.es/sma/tools/magentix2/`. First, we define our abstract argumentation framework for agent societies by extending abstract value-based argumentation frameworks [12] to work with agent societies. After that, the abstract argumentation framework proposed is instantiated by using a case-based approach to define the structure and relations of its elements. Subsequently, the reasoning process by which agents can automatically generate, select and evaluate arguments is presented.

To illustrate these concepts, let us propose the following running example, which is based on the evaluation scenario used in Section 4. We have a MAS that manages a call centre that provides customer support, where a set of agents representing technicians must reach agreements about the best solution to apply to different types of software or hardware problems reported to the centre. Therefore, to solve each problem, a group of technicians engage in an argumentation dialogue proposing their individual solutions and justifying the reasons that they have to propose them as the best solution for the problem. The *solution* that an agent proposes defines its *position* and hence, these terms are used as synonyms in this paper. At a high level of abstraction, we consider as a problem any type of conflict that agents need to solve and reach an agreement about the best solution to apply (e.g. resource allocation, classification, prediction, etc.). Also, we assume that a problem can be characterised by a set of features that describe it.

2.1. Abstract Argumentation Framework for Agent Societies

Based on Dung's abstract argumentation framework [18], we define an Argumentation Framework for an Agent Society ($AFAS$) as:

Definition 2.1 (Argumentation Framework for an Agent Society). *An argumentation framework for an agent society is a tuple* $AFAS = <A, R, S_t>$ *where:*

- A *is a set of arguments.*
- $R \subseteq A \times A$ *is an irreflexive binary attack relation on* A*.*
- S_t *is a society of agents.*

Therefore, for two arguments A and B, $attack(A, B)$ defines a *conflict* between these arguments and means that the argument A attacks the argument B. Also, in this definition we follow the approach of [17] and [7], who define an *agent society* in terms of a set of *agents* that play a set of *roles*, observe a set of *norms* and a set of *dependency relations* between roles and use a *communication language* to collaborate and reach the global objectives of the *group*. This definition is generic enough to fit most types of agent societies, such as social networks of agents or open agent organisations. Broadly speaking, it can be adapted to any open MAS where there

are norms that regulate the behaviour of agents, roles that agents play, a common language that allow agents to interact defining a set of permitted locutions and a formal semantics for each of these elements. Here, we differentiate the concept of agent society from the concept of group of agents, in the sense that we consider that the society is the entity that establishes the dependency relations between the agents, which can dynamically group together to pursue common objectives in a specific situation.

In addition, we consider that the *values* that individual agents or groups want to promote or demote and the preference orders over them have also a crucial importance in the definition of an argumentation framework for agent societies. These values could explain the reasons that an agent has to give preference to certain beliefs, objectives, actions, etc. For instance, in our running example a technician might propose a different solution for the same problem depending on whether he prefers to offer quick solutions, promoting for instance the value *efficiency*, or else high quality solutions, promoting for instance the value *accuracy*. Also, *dependency relations* between roles could imply that an agent must change or violate its value preference order. Getting back to the example, a technician of higher hierarchy could impose its values to a subordinate. Furthermore, a technician could have to adopt a certain preference order over values to be accepted in a group. These changes on value preference orders depend on the application domain and may come from changes in the system design, from the agents' adaption mechanisms, from group acceptance rules, etc. Then, if an agent changes its preferences over values, it directly affects the agent's decisions on which is the best position or argument to propose in a specific situation. Therefore, we endorse the view of [32], [44] and [12, 11], who stress the importance of the audience in determining whether an argument (e.g. for accepting or rejecting someone else's beliefs, objectives or action proposals) is persuasive or not. Thus, we have provided a formal definition of an agent society, taking into account the notion of dependency relations, values and preference orders among them:

Definition 2.2 (Agent Society). *An Agent society in a certain time t is defined as a tuple* $S_t = <$ *Ag, Rl, D, G, N, V, Roles, Dependency, Group, val,* $Valpref_Q >$ *where:*

- $Ag = \{ag_1, ag_2, ..., ag_I\}$ *is a finite set of* I *agents members of* S_t *in a certain time* t*.*
- $Rl = \{rl_1, rl_2, ..., rl_J\}$ *is a finite set of* J *roles that have been defined in* S_t*.*
- $D = \{d_1, d_2, ..., d_K\}$ *is a finite set of* K *possible dependency relations among roles defined in* S_t*.*
- $G = \{g_1, g_2, ..., g_L\}$ *is a finite set of groups that the agents of* S_t *form, where each* $g_i, 1 \leq i \leq L, g_i \in G$ *consist of a set of agents* $a_i \in Ag$ *of* S_t*.*
- N *is a finite set of norms that affect the roles that the agents play in* S_t*.*
- $V = \{v_1, v_2, ..., v_P\}$ *is a set of* P *values predefined in* S_t*.*
- $Roles : Ag \rightarrow 2^{Rl}$ *is a function that assigns an agent its roles in* S_t*.*
- $Dependency_{S_t} : \forall d \in D, <_d^{S_t} \subseteq Rl \times Rl$ *defines a reflexive, symmetric and transitive partial order relation over roles.*
- $Group : Ag \rightarrow 2^G$ *is a function that assigns an agent its groups in* S_t*.*
- $val : Ag \rightarrow V$ *is a function that assigns an agent the set of values that it has.*

- $Valpref_Q : \forall q \in Ag \cup G, <_q^{S_t} \subseteq V \times V$ *defines a reflexive, symmetric and transitive partial order relation over the values of an agent or a group.*

That is, $\forall r_1, r_2, r_3 \in R, r_1 <_d^{S_t} r_2 <_d^{S_t} r_3$ implies that r_3 has the highest rank with respect to the dependency relation d in S_t. Also, $r_1 <_d^{S_t} r_2$ and $r_2 <_d^{S_t} r_1$ implies that r_1 and r_2 have the same rank with respect to d. Finally, $\forall v_1, v_2, v_3 \in V, Valpref_{ag} = \{v_1 <_{ag}^{S_t} v_2 <_{ag}^{S_t} v_3\}$ implies that agent ag prefers value v_3 to v_2 and value v_2 to value v_1 in S_t. Similarly, $Valpref_g = \{v_1 <_g^{S_t} v_2 <_g^{S_t} v_3\}$ implies that group g prefers value v_3 to v_2 and value v_2 to value v_1 in S_t.

Now, we specialise $AFAS$ considering them for a specific agent [11], since each agent of an open MAS can have a different preference order over values. Thus, an *audience* is defined as a preference order over values. An agent-specific argumentation framework for an agent society is a tuple $AAFAS = <$ Ag, Rl, D, G, N, A, R, V, $Role$, $Dependency_{S_t}$, $Group$, $Values$, val, $Valpref_{ag_i} >$ where:

- Ag, Rl, D, G, N, A, R, V, $Dependency_{S_t}$, $Group$ and $Values$ are defined as in Definition 2.2.
- $Role(ag, a) : Ag \times A \rightarrow Rl$ is a function that assigns an agent the specific role that it plays (from its set of roles) when it has put forward a specific argument.
- $val(ag, a) : Ag \times A \rightarrow 2^V$ is a function that assigns an agent's argument the value(s) that it promotes.
- $Valpref_{ag_i} \subseteq V \times V$, defines a irreflexive, transitive and asymmetric relation $<_{ag_i}^{S_t}$ over the agent's ag_i values in the society S_t.

The aim of $AAFAS$ is to determine which agents' arguments attack other agents' arguments in an argumentation process performed in a society of agents and in each case, which argument would defeat the other. To do that, we have to consider the values that arguments promote and their preference relation, but also the dependency relations between agents. These relations could be stronger than value preferences in some cases (depending on the application domain). In our framework, we consider the dependency relations defined in [16]:

- *Power*: when an agent has to accept a request from another agent because of some predefined domination relationship between them. For instance, considering a society S_t that represents the technicians of the call centre example, a technician of high hierarchy (let us say a *manager*) has a power dependency relation over basic technicians (let us say *operators*). Thus, $Operator <_{Pow}^{S_t} Manager$.

- *Authorization*: when an agent has committed itself to another agent for a certain service and a request from the latter leads to an obligation when the conditions are met (e.g. for the duration of this service). Thus, this relation is based on the deontic notions of permission and violation, in the sense that this relation permits an agent to request services from another agent, whose refusal leads to a violation of the agreement. For instance, in S_t, $Operator <_{Auth}^{S_t} Expert$, if an operator has contracted an agreement with its company to join a team that works on a specific project, which is temporally leaded by an experienced operator that has acquired the role of *expert* and has authority to manage the team. By the agreement contracted by the operator, this operator commits himself to provide an assistance service in this project and the expert in charge of the project is permitted (authorized) to impose his decisions on the operator and the other team members.

- *Charity*: when an agent is willing to answer a request from another agent without being obliged to do so. For instance, in S_t, technicians of the same level (e.g. with the same

role in the same project) have a charity dependency relation between them by default. Thus, $Operator_i <^{S_t}_{Ch} Operator_j$, for instance.

Thus, we can now define the agent-specific defeat relation of our $AAFAS$ as:

Definition 2.3 (Defeat). *In an $AAFAS$, an agent's ag_1 argument a_1 put forward in the context of a society S_t $defeats_{ag_1}$ other agent's ag_2 argument a_2 iff*
$attack(a_1, a_2) \wedge (val(ag_1, a_1) <^{S_t}_{ag_1} val(ag_1, a_2) \notin Valpref_{ag_1}) \wedge$
$(Role(ag_1) <^{S_t}_{Pow} Role(ag_2) \vee Role(ag_1) <^{S_t}_{Auth} Role(ag_2) \notin Dependency_{S_t})$

Then, we express that the argument a_1 $defeats_{ag_1}$ from the ag_1 point of view the argument a_2 as $defeats_{ag_1}(a_1, a_2)$ if a_1 attacks a_2, ag_1 prefers the value promoted by a_1 to the value promoted by a_2 and ag_2 does not have a power or authority relation over ag_1. Also, we express the acceptability of arguments in an agent society as follows.

Definition 2.4 (Acceptability). *An argument $a_i \in A$ is $acceptable_{ag_i}(a_i)$ in a society S_t wrt a set of arguments $ARG \in A$ iff $\forall a_j \in A \wedge defeats_{ag_i}(a_j, a_i) \rightarrow \exists a_k \in ARG \wedge defeats_{ag_i}(a_k, a_j)$.*

Therefore, at the abstract level our argumentation framework defines the elements of argumentation frameworks as proposed in [34]:

- A definition for the *notion of argument* that agents use to justify their proposals. This definition takes into account the values that agents want to promote with their arguments.
- The *logical language* that agents use to represent knowledge (arguments and argumentation concepts) and engage in argumentation processes. This is an OWL-DL ontological language that allows heterogeneous agents to understand each other and communicate.
- A definition for the *concept of conflict between arguments*, which specifies when an argument attacks other different arguments.
- A definition for the *concept of defeat between arguments*, which specifies which attacks over arguments succeed, taking into account the values that arguments promote and the dependency relations among agents.
- A definition for the possible *acceptability status of arguments* during the dialogue, which classifies them as *acceptable*, *unacceptable* or *undecided* in view of their relations with other arguments [18].

At the design and implementation level, these concepts are determined by our case-based approach to instantiate the abstract argumentation framework for agent societies presented in this section. Next section provides an overview of the framework architecture.

2.2. Framework Architecture

In our argumentation framework, arguments and argumentation-related concepts are represented by using *case-based knowledge resources*. Reasoning with cases is especially suitable when there is a weak (or even unknown) domain theory, but acquiring examples encountered in practice is easy. Most argumentation systems produce arguments by applying a set of inference rules. Rule-based systems require eliciting an explicit model of the domain. In open MAS the domain is highly dynamic and the set of rules that model it is difficult to specify in advance. However, tracking the arguments that agents put forward in argumentation processes could be relatively simple. Then, these arguments can be stored as cases in a case-base and this information can be reused later to generate new arguments in view of the acquired argumentation

experience. Therefore, our framework defines a case-based reasoning process by which agents can manage arguments following different dialogue strategies. In this way, agents generate, select and evaluate arguments by making use of two types of *individual* knowledge resources that each agent has:

- **A case-base with domain-cases** that represent previous problems and their solutions. In our running example, a problem could be characterised as a tuple containing data about the actual incident occurred (e.g. OS, hardware brand, customer observations, etc.). Agents can use this knowledge resource to generate their positions in a dialogue and arguments to support them. Also, the acquisition of new domain-cases increases the knowledge of agents about the domain under discussion.

- **A case-base with argument-cases** that store previous argumentation experiences and their final outcome. In our running example, these cases could include data about the arguments interchanged between technicians to solve incidents (e.g. data about the technician that proposed the argument, data about the technician to whom the argument was addressed, their dependency relation, the solution supported by the argument and the value promoted, the acceptability status of the argument at the end of the argumentation process, etc.). Argument-cases have three main objectives: they can be used by agents 1) to generate new arguments; 2) to strategically select the best position to propose in view of the degree of acceptance that a similar position had in the past; and 3) to store the new argumentation knowledge gained in each agreement process, improving the agents' argumentation skills.

We use ontologies as the representation language for the knowledge resources of our framework. Concretely, we assume that domain-cases are instances of a domain-dependent OWL-DL ontology. Also, to represent argument-cases we have created a case-based OWL-DL argumentation ontology, called *ArgCBROnto*[1]. OWL-DL ontologies use Description Logics (DLs) to provide a common language to represent the resources that is computationally tractable, rich enough to represent different types of domain-specific and general knowledge, generic enough to represent different types of arguments and compliant with the technological standards of data and argument interchange in the Web. Although agents in open MAS are heterogeneous, by only sharing these ontologies they can *understand* the argumentation information interchanged in the system. Furthermore, this OWL-DL ontological representation enables agents to perform semantic reasoning about argumentation concepts by using, for instance, a description logics reasoner. OWL-DL allows agents to make semantic inferences about positions and arguments that are quite similar to the sort of inferences that human beings could make. In this way, agents can perform automatic reasoning with semantic knowledge in addition to the syntactic properties of cases, for instance, when retrieving similar cases from the case-bases to generate their positions and arguments in view of past experiences (see Section 2.3).

Domain-cases represent previous problems and the solution applied to them. Argument-cases are the main structure that we use to computationally represent arguments in agent societies. By using these knowledge resources, agents engaged in an argumentation dialogue can create and *justify* their positions and arguments. Finally, the agent (or agents) whose position prevails at the end of the argumentation process *wins* the dialogue and persuades the other agents to accept its position as the best solution to solve the problem at hand.

[1]The complete specification of the ArgCBROnto ontology can be found at `http://users.dsic.upv.es/~vinglada/docs`.

2.2.1. Arguments

As pointed out before, in our framework agents can generate arguments from previous cases (domain-cases and argument-cases). However, note that the fact that a proponent agent uses one or several knowledge resources to generate an argument does not imply that it has to show all this information to its opponent. The argument-cases of the agents' argumentation systems and the elements of the actual arguments that are interchanged between agents is not the same (i.e. arguments that agents interchange can be generated from domain-cases and argument-cases but it does not necessarily mean that those arguments have to include all information stored in their source cases). Thus, arguments that agents interchange are defined as tuples of the form:

Definition 2.5 (Argument). *$Arg = \{\phi, v, < S >\}$, where ϕ is the conclusion of the argument, v is the value that the agent wants to promote with it and $< S >$ is a set of elements that support the argument (support set).*

Definition 2.6 (Support Set). $S = <$ {*premises*}, {*domainCases*}, {*argumentCases*}, {*distinguishingPremises*}, {*counterExamples*} $>$

This *support set* can consist of different elements, depending on the argument purpose. In our framework, agents can generate two types of arguments: *support arguments*, to justify their positions and *attack arguments* to attack the positions and arguments of other agents. On one hand, if the argument is a support argument, the support set is the set of features (*premises*) that represent the context of the domain where the argument has been proposed (those premises that match the problem to solve and other extra premises that do not appear in the description of this problem but that have been also considered to draw the conclusion of the argument) and optionally, any knowledge resource used by the proponent to generate the argument (*domain-cases* and *argument-cases*). Also, a support argument promotes the value promoted by the position that it justifies. In our framework, we assume that an attack argument promotes the value promoted by the position that it tries to defend (if an agent has generated it to rebut an attack on its support argument) or else, an attack argument promotes the agent's most preferred value over the set of values pre-defined in the system (if an agent has generated it to attack the position of other agent). On the other hand, if the argument is an attack argument, the support set can include any of the allowed attacks in our framework (*distinguishing premises* or *counter-examples*), as proposed in [12].

Let us assume that we have a set of cases denoted as $C = \{c_1, c_2, ..., c_n\}$, a set of premises denoted as $F = \{x_1, x_2, ..., x_m\}$, a problem to solve denoted as P (characterised by a subset of the premises of F), a function $value_c(x)$ that returns the value of a premise $x \in F$ in a case $c \in C$ (i.e. the actual data of that premise, do not confuse with the notion of value promoted by arguments), a function *acceptable(c)* that returns *true* if the case c was deemed acceptable, and a function *conclusion(c)* that returns the conclusion of the case c (i.e. the solution promoted by the case).

Definition 2.7 (Distinguishing Premise). *A distinguishing premise x_i with respect to a problem P between two cases $c_1, c_2 \in C$ is defined as: $\exists x_i \in c_1 \wedge \nexists x_i \in P \mid \exists x_i \in c_2 \wedge value_{c_1}(x_i) \neq value_{c_2}(x_i)$ or else, $\exists x_i \in c_1 \wedge \exists x_i \in P \mid value_{c_1}(x_i) = value_P(x_i) \wedge \nexists x_i \in c_2$, where $P \subseteq F$, $x_i \in F$ and $c_1, c_2 \in C$.*

That is a premise that does not appear in the description of the problem to solve and has different values for two cases or a premise that appears in the problem description and does not appear in one of the cases.

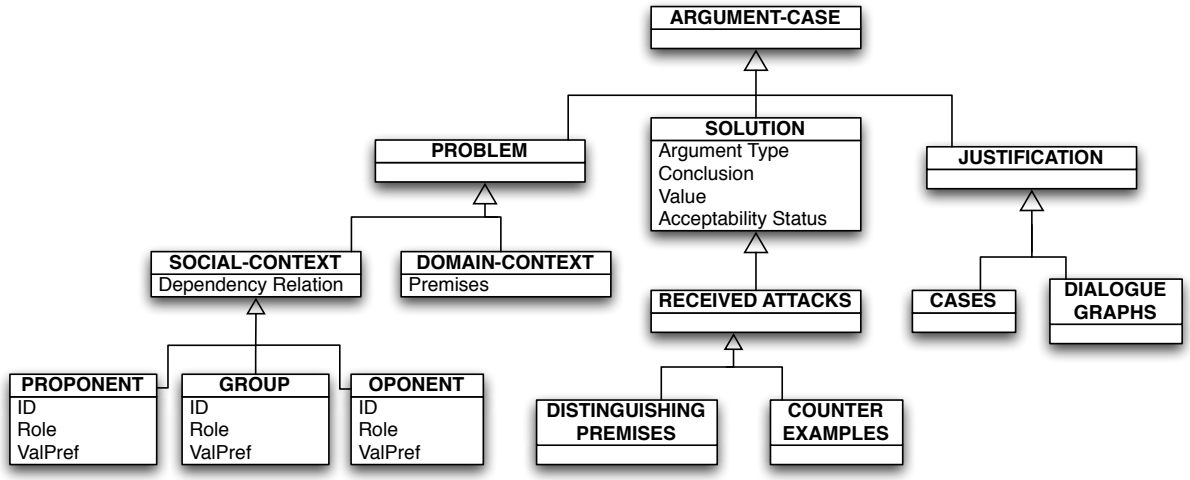

Figure 1: Structure of an Argument-Case

Definition 2.8 (Counter-Example). *A counter-example for a case $c_1 \in C$ with respect to a problem P is another case $c_2 \in C$ such that: $acceptable(c_2) \wedge \forall x_i \in c_2 \cap P / value_{c_2}(x_i) = value_P(x_i) \wedge \forall x_i \in c_1 / (\exists x_i \in c_2 \wedge value_{x_i}(c_2) = value_{x_i}(c_1)) \wedge conclusion(c_2) \neq conclusion(c_1)$*

That is, a counter-example for a case is a previous case (i.e. domain-case or an argument-case that was deemed acceptable), where the problem description of the counter-example matches the current problem to solve and also subsumes the problem description of the case, but proposing a different solution.

2.2.2. Domain-Cases

As domain-cases represent previous problem-solving experiences in a specific domain, their structure is determined by the characteristics of this domain. For instance, in the call centre example, domain-cases will store the features that characterise a problem that was solved in the call-centre and also such features that characterise the solution applied to this problem.

2.2.3. Argument-Cases

Argument-cases structure is generic and domain-independent. Figure 1 shows the generic structure of an argument-case.

Argument-cases have the three possible types of components that usual cases of CBR systems have: the description of the state of the world when the case was stored (*Problem*); the solution of the case (*Conclusion*); and the explanation of the process that gave rise to this conclusion (*Justification*).

The problem description has a *domain context* that consists of the *premises* that characterise the argument. In addition, if we want to store an argument and use it to generate a persuasive argument in the future, the features that characterise its *social context* must also be kept. The social context of the argument-case includes information about the *proponent* and the *opponent* of the argument and about their *group*. Moreover, we also store the preferences (*ValPref*) of each agent or group over the set of possible *values* that arguments can promote (pre-defined in the system). Finally, the *dependency relation* between the proponent's and the opponent's roles is also stored.

In the solution part, the *conclusion* of the case, the *value* promoted, and the *acceptability status* of the argument at the end of the dialogue are stored. The acceptability status shows if

the argument was deemed *acceptable*, *unacceptable* or *undecided* in view of the other arguments that were put forward in the agreement process. Therefore, an argument is deemed acceptable if it remains undefeated at the end of the argumentation dialogue, unacceptable if it was defeated during the dialogue and undecided if its acceptability status cannot be determined with the current information about the dialogue. In our framework, the acceptance of an argument is determined by the defeat relation presented in Definition 2.3 of Section 2.1, which determines which argument prevails when an argument attacks other argument, taking into account the agents that have proposed these arguments, their preferences over values and their dependency relation. In addition, the conclusion part includes information about the possible *attacks* that the argument received during the process. These attacks could represent the justification for an argument to be deemed unacceptable or else reinforce the persuasive power of an argument that, despite being attacked, was finally accepted. Concretely, arguments in our framework can be attacked by putting forward *distinguishing premises* or *counter-examples* to them.

Finally, the justification part of an argument-case stores the information about the knowledge resources that were used to generate the argument represented by the argument-case (the set of domain-cases and argument-cases). In addition, the justification of each argument-case has a *dialogue-graph* (or several) associated, which represents the sequence of arguments that form the *dialogue structure* where the argument was proposed. In this way, the complete conversation is stored as a directed graph that links argument-cases that represent the arguments of the dialogue. The longest path from the fist argument to the last argument stored in the dialogue graph constitutes the *dialogue depth*. This graph can be used later to improve the efficiency in an argumentation dialogue in view of similar dialogues that were held in the past. For instance, long dialogues can be prematurely interrupted if they have a high resemblance with previous dialogues that did not reach an agreement in the past or the duration of a current dialogue can be estimated by using the dialogue depth of similar past dialogues.

2.2.4. Examples

To illustrate the knowledge resources of our framework in the call centre running example (see Section 4 for further details of this example), let us propose the following examples. Figure 2 shows how two different operators, *Heras* and *Navarro*, can generate different positions by retrieving and reusing the solutions of different domain-cases from their domain-cases case-bases (*CH* and *CN* respectively). Also, note that these domain-cases are counter-examples for each other, since they share the same problem description, which also matches the description of the current problem to solve, but proposing different solutions. Furthermore, their premise *"Model"* is a distinguishing premise for each other, since it does not appear in the description of the current problem to solve and stores different data for the two cases. Note that in the call centre example we assume that domain-cases also store the value promoted by the solution that they propose. In other different application scenarios of our case-based argumentation framework this assumption could not hold and, for instance, we may assume that all positions generated by an agent promote its most preferred value.

Figure 3 shows an argument-case that represents the argument that an operator *Heras* sent to an operator *Navarro*, proposing her position to solve an incident received by the centre. This incident reports an error in a HP computer with Windows XP installed, which reboots automatically at random. Both operators belong to the group *Software Support*, which does not impose any specific value preference order over its members and have a *charity* dependency relation between them. As shown in the figure, the operator *Heras* prefers to promote the value *efficiency* of the call centre and thus, proposes a complete reinstallation to avoid the waste of time that could entail to investigate the actual cause of the incident, which would promote the value of *accuracy*, for instance. In the justification part, we can see that the operator *Heras*

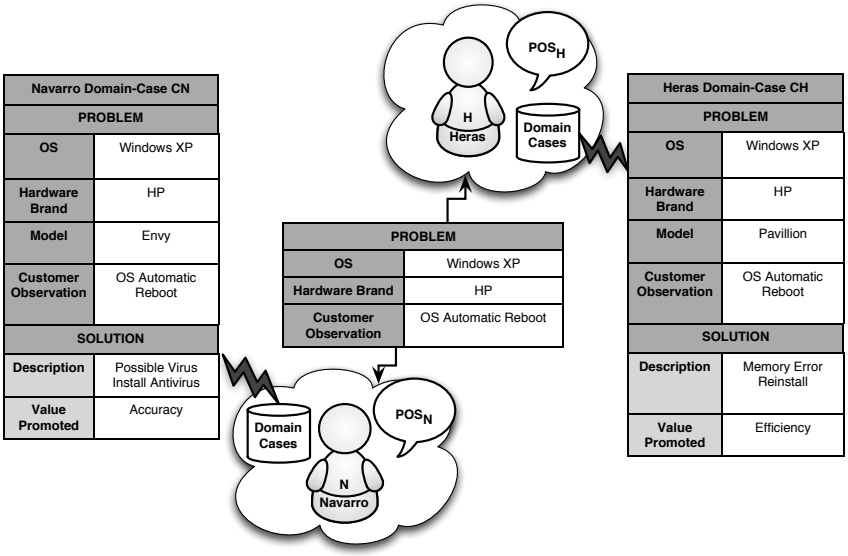

Figure 2: Structure of Domain-Cases in the Call Centre Example

generated her position by reusing the solution applied in the past to the similar case *CH*. Also, although the argument received a counter-example attack with a case *CN*, let us assume that the operator *Heras* was able to rebut the attack and hence, her argument remained acceptable at the end of the argumentation dialogue (as shown in the argument-case acceptability status). Finally, the structure that links the complete set of arguments that were put forward in the argumentation dialogue to agree a solution for the incident could be checked at the dialogue graph *DH*.

Finally, let us denote as *p1*, *p2*, *p3* and *p4* the premises that describe the domain-context of the argument-case shown in Figure 3 (the ones that characterise the OS, hardware brand, model and customer observation, respectively) and as $CONC_H$ the conclusion of this case (which entails the solution "Memory Error. Reinstall"). Thus, the argument-case of Figure 3 could represent the following support argument Arg_H proposed by the operator *Heras* to justify its position by showing the domain-case CH, which was used to generate this position:

$$Arg_H = \{CONC_H, Efficiency, < \{p1, p2, p3, p4\}, CH >\}$$

In addition, let us assume that the operator *Navarro* was the agent who generated the attack that received this argument (and that is also represented in the argument-case of Figure 3). This attack puts forward the counter-example CN, which for the same characterisation of the problem that CH represents, proposes an alternative solution $CONC_N$ ("Possible Virus. Install Antivirus", for instance). Assuming that the symbol $\sim$ stands for the logical negation of a proposition and that $CONC_N$ implies $\sim CONC_H$, the following attack argument Arg_N could represent the attack to Arg_H:

$$Arg_N = \{\sim CONC_H, Accuracy, < \{p1, p2, p3, p4\}, CN >\}$$

2.3. Reasoning Process

As explained before, domain-cases and argument-cases are used in our framework as knowledge resources that agents use to perform a reasoning process to generate, select and evaluate

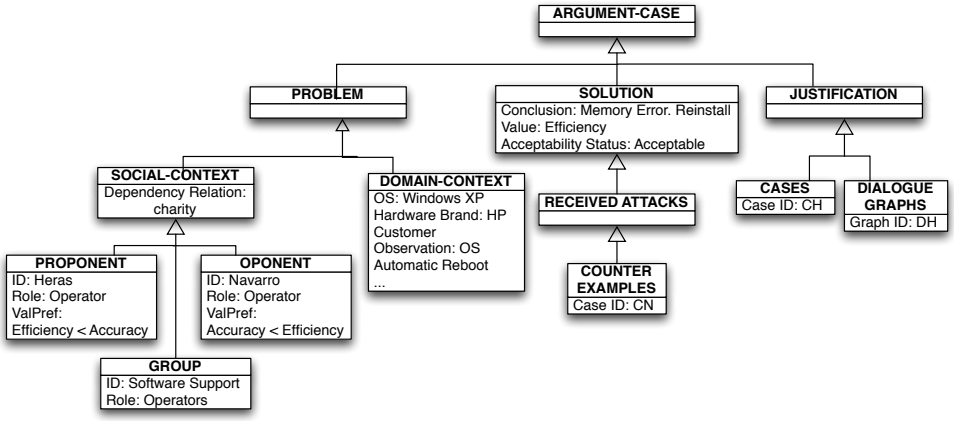

Figure 3: Structure of an Argument-Case in the Call Centre Example

positions and arguments. Agents generate their positions and arguments by performing a CBR cycle over their domain-cases case-base, selecting such domain-cases that are similar to the current problem to solve (i.e. their problem description matches the description of the current problem to solve) and adapting the solution applied to these cases to fit the current problem. Depending on the set of knowledge resources used, agents can generate different positions and arguments. Among the list of potential positions or arguments generated, agents select the most suitable one to propose by selecting those ones that promote their most preferred value and then performing a CBR cycle over their argument-cases case-base. In this way, agents can retrieve similar arguments used in the past to support their positions or attack other agents' positions and arguments. With them, agents can follow different tactics to select the best position or argument to put forward in the current step of the argumentation dialogue. These tactics consist of assigning more or less weight to the parameters of an argumentation *support factor* calculated for each position and argument generated (see Section 3). Therefore, positions and arguments are sorted by the agents' value preferences and their expected suitability. Similarly, when agents have to evaluate other agents' positions and arguments in view of their owns, they perform a CBR cycle over their domain-cases case-base to generate support or attack arguments. Then, they perform a CBR cycle over their argument-cases case-base to select the best argument to propose from the list of potential candidates. Finally, at the end of the process, agents learn the final solution applied to solve the problem in their domain-cases case-base and the arguments generated during the process in their argument-cases case-base.

The support factor estimates how suitable a current position or argument is in view of the suitability of similar arguments (to support or attack similar positions or arguments) put forward in previous argumentation dialogues, which are stored in the agent's case-base of argument-cases. In this way, the agent can select the most suitable position or argument to propose next. The support factor is computed by a linear combination of several parameters, which are calculated by using the following formulas. In them, $argC$ is the subset of argument-cases from the whole set arg stored in the agent case-base with the same problem description and conclusion as the current argument; $argAccC$ are those in $argC$ that were deemed acceptable at the end of the dialogue; $argAccCAtt$ are those in $argAccC$ that were attacked; att is the number of attacks received by similar argument-cases in the past; $minAtt$ and $maxAtt$ are the minimum and maximum number of attacks received by any argument-case in $argAccCAtt$; n is the depth from the node representing a similar argument-case retrieved to the node representing the conclusion in the dialogue graphs associated with it; $minS$ and $maxS$ are the minimum and maximum

number of steps from any argument-case in $argC$ to the last node of its dialogue graph and; kr is the number of knowledge resources that were used to generate each similar argument-case retrieved; $minKr$ and $maxKr$ are the minimum and maximum number of knowledge resources used to generate any argument-case in $argC$. Also, we assume that the modifier # represents the number of elements of a set.

- **Persuasiveness Degree (PD):** is a value that represents the expected persuasive power of an argument by checking how persuasive an argument-case with the same problem description and conclusion was in the past. To compute this degree, the number $\#argAccC$ of argument-cases that were deemed acceptable out of the total number of retrieved argument-cases $\#argC$ with the same problem description and conclusion as the current argument is calculated:

$$PD = \begin{cases} 0, & if\ \#argC = \emptyset \\ \dfrac{\#argAccC}{\#argC}, & otherwise \end{cases} \quad (1)$$

  with $\#argAccC$, $\#argC \in \mathbb{N}$ and $PD \in [0, 1]$, from less to more persuasive power.

- **Support Degree (SD):** is a value that provides an estimation of the probability that the conclusion of the current argument was acceptable at the end of the dialogue. It is based on the number of argument-cases $\#argAccC$ with the same problem description and conclusion that where deemed acceptable out of the total number of argument-cases arg retrieved.

$$SD = \begin{cases} 0, & if\ \#arg = \emptyset \\ \dfrac{\#argAccC}{\#arg}, & otherwise \end{cases} \quad (2)$$

  with $\#argAccC, \#arg \in \mathbb{N}$ and $SD \in [0, 1]$ from less to more support degree.

- **Risk Degree (RD):** is a value that estimates the risk for an argument to be attacked in view of the attacks received for an argument(s) with the same problem description and conclusion in the past. It is based on the number of argument-cases $\#argAccCAtt$ that were attacked out of the total number of $\#argAccC$ argument-cases with the same problem description and conclusion retrieved that were deemed acceptable.

$$RD = \begin{cases} 0, & if\ \#argAccC = \emptyset \\ \dfrac{\#argAccCAtt}{\#argAccC}, & otherwise \end{cases} \quad (3)$$

  with $\#argAccCAtt$, $\#argAccC \in \mathbb{N}$ and $RD \in [0, 1]$, from less to more risk of attack.

- **Attack degree (AD):** is a value that provides an estimation of the number of attacks att received by a similar argument(s) in the past. To compute this degree, the set of argument-cases with the same problem description that were deemed acceptable is retrieved. Then, this set is separated into several subsets, one for each different conclusion that these argument-cases entail. The sets whose conclusion match with the conclusions of the arguments to assess are considered, while the other sets are discarded. Thus, we have a set of argument-cases for each different potential argument (and its associated

conclusion) that we want to evaluate. For each argument-case in each set, the number of attacks received is computed (the number of distinguishing premises and counter-examples received). Then, for each set of argument-cases, the average number of attacks received is computed. The attack degree of each argument is calculated by a linear transformation:

$$AD = \begin{cases} 0, & if\ maxAtt = minAtt \\ \dfrac{att - minAtt}{maxAtt - minAtt}, & otherwise \end{cases} \quad (4)$$

with $minAtt$, $maxAtt$, $att \in \mathbb{N}$ and $AD \in [0, 1]$ from less to more degree of attack.

- **Efficiency degree (ED):** is a value that provides an estimation of the number of steps that it took to reach an agreement posing a similar argument in the past. It is based on the depth n from the node representing a similar argument-case retrieved to the node representing the conclusion in the dialogue graphs associated with it. To compute this degree, the same process to create the subsets of argument-cases as in the above degree is performed. Then, for each argument-case in each subset, the number of dialogue steps from the node that represents this argument-case to the end of the dialogue is computed. Also, the average number of steps per subset is calculated. Finally, the efficiency degree of each argument is calculated by a linear transformation:

$$ED = \begin{cases} 0, & if\ maxS = minS \\ 1 - \dfrac{n - minS}{maxS - minS}, & otherwise \end{cases} \quad (5)$$

with $minS$, $maxS$, $n \in \mathbb{N}$ and $ED \in [0, 1]$ from less to more efficiency.

- **Explanatory Power (EP):** is a value that represents the number of pieces of information that each argument covers. It is based on the number kr of knowledge resources were used to generate each similar argument-case retrieved. To compute this number, the same process to create the subsets of argument-cases as in the above degrees is performed. Then, for each argument-case in each set, the number of knowledge resources in the justification part is computed (the number of domain-cases and argument-cases). Then, for each set of argument-cases, the average number of knowledge resources used is computed. The explanatory power of each argument is calculated by a linear transformation:

$$EP = \begin{cases} 0, & if\ maxKr = minKr \\ \dfrac{kr - minKr}{maxKr - minKr}, & otherwise \end{cases} \quad (6)$$

with $minKr$, $maxKr$, $kr \in \mathbb{N}$ and $EP \in [0, 1]$ from less to more explanatory power.

The selection of these specific parameters to estimate the support factor of a position or argument has been determined by the nature of the elements of our argument-cases. Thus, the persuasiveness and support degrees take into account the acceptability status stored in the argument-cases, the attack and risk degrees look at the attacks received by the argument that an argument-case represents, the efficiency degree makes use of the dialogue graphs stored in the argument-cases and the explanatory power computes the number of justification elements that argument-cases have. Therefore, the support factor is computed by using the following formula:

$$SF = w_{PD} * PD + w_{SD} * SD + w_{RD} * (1 - RD) + w_{AD} * (1 - AD) + w_{ED} * ED + w_{EP} * EP \quad (7)$$

where $w_i \in [0, 1], \sum w_i = 1$ are weight values that allow the agent to give more or less importance to each parameter of the support factor. Finally, positions and arguments are ordered from more to less suitability to be proposed by following the equation:

$$Suitability = w_{SimD} * SimD + w_{SF} * SF \quad (8)$$

where $w_i \in [0, 1], \sum w_i = 1$ are weight values that allow the agent to give more or less importance to the similarity degree with the problem to solve ($SimD$) or the support factor. Finally, the most suitable position or argument that promotes the proponent agent most preferred value is selected as the one that it is going to propose and defend first.

Algorithm 2.1 positionGenerationAndSelection

Require: ProblemDescription, w_{SimD}, w_{PD}, w_{SD}, w_{RD}, w_{AD}, w_{ED}, w_{EP} *//The description of the problem to solve and a set of weights to give more or less importance to each element of the similarity degree and the support factor*

```
positions = ∅
argumentCases = ∅
SimD = ∅
SF = ∅
selectedPositions = ∅
positions = generatePositions(ProblemDescription, n)
for [i = 1;i ≤ lenght(positions);i++] do
    argumentCases[i] = retrieveArgumentCases(ProblemDescription, positions[i])
    SimD[i] = computeSimilarityDegree(positions[i], w_SimD)
end for
for [i = 1;i ≤ lenght(argumentCases);i++] do
    SF[i] = computeSF(ProblemDescription, argumentCases, w_PD, w_SD, w_RD, w_AD, w_ED, w_EP)
end for
selectedPositions = selectPosition(positions, SimD, SF)
Return mostSuitable(selectedPositions)
```

To illustrate part of the case-based reasoning process proposed in our framework, Algorithm 2.1 shows the pseudocode of the algorithm that implements the generation and selection of positions. The whole process to manage positions and arguments is detailed in [19, Chapter 3]. In the algorithm, the function *generatePositions* generates the n first positions (or all possible positions if they are less than n) by retrieving a list of past similar domain-cases and reusing their solutions (actually, the function uses the Euclidean distance to retrieve similar cases and then copies the solutions of these cases, although alternative retrieving and reusing techniques could be considered); *retrieveArgumentCases* is a function that retrieves for each position the list of argument-cases that represent arguments that were used to attack or support a similar position with a similar social context (again using the Euclidean distance); *computeSimilarity-Degree* is a function that computes a *degree of similarity (SimD)* of each position generated with regard to the problem to solve (by using the Euclidean distance or an algorithm that computes the semantic distance between ontological concepts); *computeSF*, is a function that computes the *support factor (SF)* of each position by using the formula explained before; *selectPosition* is a domain-dependent function that orders the set of positions from more to less suitable with respect to some domain-dependent criteria, taking into account their similarity degree and support factor; and *mostSuitable* is a domain-dependent function that returns the most suitable position to solve the problem.

For instance, in the evaluation example presented in Section 4 the *selectPosition* function is implemented in the following way: the first step for selecting positions is sorting them in subsets, taking into account the value that promotes each one. Thus, the functions assigns each set a *Suitability Level (SL)*. Positions that promote the value most preferred by the agent will be labelled with suitability level 1, positions that promote the second most preferred value will be labelled with suitability level 2 and so on. After that, positions will be ordered in each level by a linear combination between their similarity degree with the problem to solve and their support factor. Also, the *mostSuitable* function is a simple function that selects the first element of the ordered list computed by the *selectPosition* function.

The next section explains how an agent can make use of the elements of our case-based argumentation framework to devise heuristic dialogue strategies and persuade other agents participating in the argumentation dialogue to accept its position as the best solution to apply.

3. Case-based Dialogue Strategies

Agents of our argumentation framework follow a case-based reasoning process to generate arguments, to select the best argument to put forward taking into account their social context and to evaluate arguments in view of other arguments proposed in the dialogue. Also, they use a dialogue protocol to exchange information and engage in the argumentation process. In each step of an argumentation dialogue an agent can choose a specific locution to put forward and a content for it. The mechanism that agents follow to make such decisions is known as *dialogue strategy*.

Our dialogue protocol includes locutions for opening a new dialogue (e.g. *open_dialogue*), for entering in the dialogue (e.g. *enter_dialogue*), for proposing positions (e.g. *propose*), for challenging positions (e.g. *why?*), for advancing arguments (e.g *assert*), for attacking arguments (e.g. *attack*), etc. The complete set of locutions and dialogue rules that agents use can be consulted in [19, Chapter 4]. Here, we focus on the case-based dialogue strategies that agents can follow during the argumentation process. Thus, assuming that L represents the set of available locutions of the protocol that agents use to communicate, let us suppose that the function *Replies* returns for each locution the set of locutions that constitute legal replies to it:

$$Replies : L \rightarrow 2^L$$

Then, assuming that D represents the set of well-formed formulae in the logical language used in the framework to represent arguments and knowledge resources, the function *Content* returns for a given locution, the set of possible contents:

$$Content : L \rightarrow 2^D$$

In each step of the argumentation dialogue, agents exchange moves.

Definition 3.1 (Move). *A move is a pair (*l*, φ), where $l \in L$ and $\varphi \in$ Content(l).*

Thus, the strategy problem is formalised as in [3]:

Definition 3.2 (Strategy Problem). *Let (*l*, φ) be the current move in a dialogue. What is the next move (*l'*, φ') to utter such that* l' $\in$ *Replies(*l*)?*

The answer for this question implies to find the best locution and content that the agent can utter in each step of the dialogue, given the attitude (profile) of the agent and its knowledge resources. Therefore, a dialogue strategy is formally defined as follows:

Definition 3.3 (Dialogue Strategy). *A dialogue strategy is defined as a function S:* $L \times D \rightarrow L \times D$ *where (*l, φ*)* $\in L \times D$.

Given a move *m=(l,* φ*), S(m) = m'* such that *m' = (l',* φ*')* is the best move that an agent can utter in the next step of the dialogue taking into account its profile and knowledge. In our case-based argumentation framework, agents select the best locution to make depending on their *profile* and the content of this locution depending on the knowledge that they have in their knowledge resources and the *tactic* that they follow to argue. Therefore, we present our concept of dialogue strategies as a combination of the agents' profile and tactic. The following sections present these elements in detail.

3.1. Agent Profiles

The profile of an agent defines its attitude towards the arguments put forward by itself and other different agents during the dialogue. Therefore, the profile of the agent determines the strategical aspect of deciding which is the best locution to put forward at each step of the dialogue. In our framework, we consider the following agent profiles, as proposed in [6]:

- Agreeable: accept whenever possible.
- Disagreeable: only accept when there is no reason not to.
- Open-minded: only attack when necessary.
- Argumentative: attack whenever possible.
- Elephant's child: challenge whenever possible.

Agent Profile	**Accept**			**Challenge**			
	ASP	**APL**	**TAP**	**CSP**	**CPL**	**CDP**	**CP∅**
Agreeable	√	√					
Disagreeable	√						
Open-Minded			√			√	
Argumentative					√	√	
Elephant's Child				√	√	√	√

Table 1: Dialogue decisions that agents make depending on their profile

Table 1 summarises the behaviour of each agent profile when the rules of the dialogue allow for advancing several different positions at certain step. The table shows how the profile of agents determine their decisions about the best locution to put forward in each step of the dialogue. The legends of the columns of Table 1 are the following:

- **ASP (Accept Same Position):** Accept the position of a peer that matches the agent's current position and do not argue with it.
- **APL (Accept Position in List):** Accept the position of a peer that is in the agent's list of potential positions (although not ranked as the most suitable position to propose) and do not argue with it.
- **TAP (Take Attacked Position):** When the agent has attacked the position of another agent and the latter wins the debate, accept its position an change the agent's current position for the accepted position.

- **CSP (Challenge Same Position):** Challenge the position of a peer that has proposed the same position as the agent's one.
- **CPL (Challenge Position in List):** Challenge the position of a peer that has proposed a position that is in the agent's list of potential positions (although not ranked as the most suitable position to propose).
- **CDP (Challenge Different Position):** Challenge the position of peer that has proposed a different position (and this position is not in the agent's list of potential positions to propose).
- **CP$\emptyset$ (Challenge other Positions if no position can be generated).**

If an agent *accepts* the position of another agent, it makes a vote for this position. In case of draw at the end of the dialogue, the position with the largest number of votes may be selected as the winning position. This is a design decision and depends on the application domain. However, the fact that an agent accepts the position of another agent does not necessarily mean that this agent has to *take* this position and change its own position by the accepted position.

An *agreeable* agent will deem positions and arguments from peers (other agents that it has a charity dependency relation with them) as acceptable whenever it is possible. This means that it will not challenge any position that is in the list of its potential positions (even if it is not ranked as the most suitable) and it will accept any argument from a peer if it does not have a counter-argument or a distinguishing premise that attack it. Therefore, if the agent cannot rebut an argument from a peer, it will accept it and also its associated position. Agreeable agents do not challenge positions of other agents, but just try to defend theirs if attacked. In the case that an agreeable agent cannot generate a position, it does not participate in the dialogue.

A *disagreeable* agent will deem the position of a peer as acceptable if it is ranked first in its list of potential positions. Regarding arguments, this type of agent will try to generate an attack to any argument that it receives from other agents. If it is not able to generate such an attack, the agent will accept the argument of its peer, but it still will not accept the peer's position. Disagreeable agents do not challenge positions of other agents, but just try to defend theirs if attacked. In the case that a disagreeable agent cannot generate a position, it does not participate in the dialogue.

An *open-minded* agent does not accept positions from other agents by default, but it challenges different positions of other peers. Also, it waits for challenges from other peers and will try to rebut their attack arguments. If a peer wins the discussion, this type of agent accepts its argument and changes its own position by the peer's position even when its own position is not the target of debate. If it cannot generate positions, it does not engage in the dialogue.

An *argumentative* agent will not deem any position from a peer as acceptable by default. This type of agent will challenge positions of other peers when they are different from its position, even if they appear in its list of potential positions to propose. Also, it will try to generate an answer for any attack that it receives, but opposite to open-minded agents, argumentative agents do not accept the position of the peer that generated the attack if the last wins the debate. If an argumentative agent cannot generate positions, it will not participate in the dialogue.

An *elephant's child* agent will always challenge the positions of other peers (even if they have proposed the same position as it or if it cannot generate positions). If it can generate attacks, it will put them forward to rebut the arguments of other agents, but if they win the debate, this type of agent does not accept their positions. In fact, the only way an elephant's child agent accepts the position of another agent is when it challenges this position, the attacked agent provides a support argument, the elephant's child agent attacks this argument and the attacked agent wins the debate.

Independently of their profile, agents will accept arguments from other agents that have a power or authorization dependency relation over them. Recall that in any case the acceptance of an argument is subjected to the defeat relation defined in the argumentation framework and presented in Definition 2.3 of Section 2.1, which determines which argument prevails when an argument attacks other argument, taking into account the agents that have proposed these arguments, their preferences over values and their dependency relation. As pointed out before, depending on its profile, the agent will choose the next locution to put forward in the dialogue. For instance, let us assume that an agent a_i has proposed a position q to solve the problem p under discussion, an agreeable agent a_j has entered in the dialogue and proposed a position q' and a_i has challenged the position of a_j. In this case, the agreeable agent a_j will try to generate a support argument for its position by searching its domain and argument-cases case-bases. Then, among the potential arguments that a_j could generate, it has to select one to support the position. This implies to select the content of the locution to assert the support argument. To make this selection, agents can use several tactics. The next section shows the tactics that agents of our case-based argumentation framework can follow.

3.2. Dialogue Tactics

Once an agent has decided (by using its profile and knowledge resources) which is the best locution to put forward in the next step of the dialogue, it has to select the content for this locution depending on its tactic. This content consists of the actual argument that the agent will propose, from the set of potential candidates generated by using its knowledge resources. Therefore, the tactic that an agent follows determines the strategical aspect of deciding which is the best argument for the next locution to put forward in the argumentation dialogue.

From our point of view, a tactic consists of assigning more or less weight to the elements of the *support factor* used to select positions and arguments in the agents' reasoning process. As pointed out in Section 2, the support factor estimates how suitable a current argument is in view of the suitability of similar arguments put forward in previous argumentation dialogues, which are stored in the agent's case-base of argument-cases. Therefore, our tactics are built on the knowledge that agents gained from previous argumentation dialogues. In this way, the agent can select the most suitable argument to propose next (the most suitable content of the locution to state). The importance (weight) that an agent gives to each parameter of the support factor determines the tactic that the agent is following. Thus, we consider the following tactics to select the best argument to fit the content of the locution to put forward in the next step of the argumentation dialogue:

- Persuasive Tactic: the agent selects such arguments in which similar argument-cases were more persuasive in the past (assigns more weight to the persuasiveness degree).
- Maximise-Support Tactic: the agent selects such arguments that have higher probability of being accepted at the end of the dialogue (assigns more weight to the support degree).
- Minimise-Risk Strategy: the agent selects such arguments that have a lower probability of being attacked (assigns more weight to the risk degree).
- Minimise-Attack Tactic: the agent selects such arguments that have received a lower number of attacks in the past (assigns more weight to the attack degree).
- Maximise-Efficiency Tactic: the agent selects such arguments that lead to shorter argumentation dialogues (assigns more weight to the efficiency degree).

- Explanatory Tactic: the agent selects such arguments that cover a bigger number of cases. That is, such arguments that are similar to argument-cases that have more justification elements (assigns more weight to the explanatory power).

As pointed out before, the dialogue strategy that an agent follows consists of the combination of its profile and tactic. For instance, combing the argumentative profile with the above tactics, we could have the following argumentation strategies:

1. Persuasive Strategy: the agent argues whenever possible, selecting such arguments expected to be more persuasive.
2. Maximise-Support Strategy: the agent argues whenever possible, selecting such arguments expected to have higher probability of being accepted at the end of the dialogue.
3. Minimise-Risk Strategy: the agent argues whenever possible, selecting such arguments expected to have lower probability of being attacked.
4. Minimise-Attack Strategy: the agent argues whenever possible, selecting such arguments expected to receive a lower number of attacks.
5. Maximise-Efficiency Strategy: the agent argues whenever possible, selecting such arguments expected to lead to shorter dialogues.
6. Explanatory Strategy: the agent argues whenever possible, selecting such arguments expected to have more elements to be justified.

Thus, different strategies can be more or less suitable depending on the profile of the other agents that participate in the dialogue, the tactics that they follow and their available knowledge resources. Now, the following section tests several dialogue strategies in a real application domain and analyses the results achieved.

4. Evaluation

In this section, we evaluate the performance of the dialogue strategies proposed in our case-based argumentation framework by running a set of empirical tests. With this objective, the framework has been implemented in the domain of a customer support application. Concretely, we consider a society of agents that act on behalf of a group of technicians that must solve problems in a Technology Management Centre (TMC). TMCs are entities which control every process implicated in the provision of technological and customer support services to private or public organisations. Usually, TMCs are managed by a private company that communicates with its customers via a call centre. These kinds of centres allow customers to obtain general information, purchase products or lodge a complaint. They can also efficiently communicate public administrations with citizens. In a call centre, there are a number of technicians attending to a big amount of calls with different objectives (e.g. sales, marketing, customer service, technical support and any business or administration activity). The call centre technicians have computers provided with a helpdesk software and phone terminals connected to a telephone switchboard that manages and balances the calls among technicians. The current implementation is based on previous work that deployed a case-based multi-agent system in a real TCM [22]. This system was implemented and is currently used by the TCM company. In the original implementation, agents were allowed to use their case-bases to provide experience-based customer support. In this work, the original system has been enhanced by allowing agents to argue about the best way of solving the incidents that the call centre receives.

Therefore, we consider a society composed by call-centre technicians with three possible roles, *operator*, *expert* and *manager*. Operators form groups that must solve the problems that the call centre receives. Experts are specialised operators that have case-bases with knowledge

about the suitable solutions to provide for specific problems and hence, are authorised to impose their opinions over operators. Managers are the administrators of the system and can take the final decision in case of conflict, for instance, if the agreement is not reached. Therefore, the dependency relations in this society establish that managers have a *power* relation over experts and operators, that experts have an *authorization* relation over operators and that technicians with the same role have a *charity* relation among them.

In this application domain we assume that each technician has a helpdesk application to manage the big amount of information that processes the call centre. The basic functions of this helpdesk are the following:

- To register the ticket information: customer data, entry channel and related project, which identifies the customer and the specific service that is being provided.
- To track each ticket and to scale it from one technician to a more specialised one or to a different technician in the same level.
- To warn when the maximum time to solve an incident is about to expire.
- To provide a solution for the ticket. This means to generate an own position or to ask for help to the members of a group.

In addition, this helpdesk would implement an argumentation module to solve each ticket as proposed in our framework. Hence, we assume the complex case where a ticket must be solved by a group of agents representing technicians that argue to reach an agreement over the best solution to apply. Each agent has its own knowledge resources (acceded via his helpdesk) to generate a solution for the ticket. Figure 4 shows the architecture of the system developed. As shown, the argumentation module of each agent includes a Domain-CBR engine that makes queries to its domain-cases case-base and an Argumentation-CBR engine that makes queries to its argument-cases case-base. The system has been implemented by using the *Magentix2* agent platform[2]. Magentix2 is an agent platform that provides new services and tools that allow for the secure and optimised management of open MAS.

The data-flow for the argumentation dialogue to solve each ticket is the following:

1. The system presents a group of technicians with a new ticket to solve.
2. An agent, called the initiator agent, opens a new argumentation dialogue with a locution *open_dialogue*.
3. Technicians enter in the dialogue by using the locution *enter_dialogue*.
4. If possible, each technician generates his own position by using the argumentation module and propose it with the locution *propose*. This module supports the argumentation framework proposed in this paper. All technicians that are willing to participate in the argumentation process are aware of the positions proposed in each moment.
5. The technicians argue to reach an agreement over the most suitable solution by following a persuasion dialogue controlled by the dialogue game protocol proposed in [19, Chapter 4]. The dialogue proceeds as a set of parallel dialogues between technicians. Therefore, one technician can only argue with another at the same time. In each parallel dialogue, two technicians take turns to challenge the positions of the other technician with the locution *why?*, to assert arguments to justify their positions with the locution *assert* or to attack the other technician's positions and arguments with the locution *attack*. If a

[2]http://users.dsic.upv.es/grupos/ia/sma/tools/magentix2/index.php

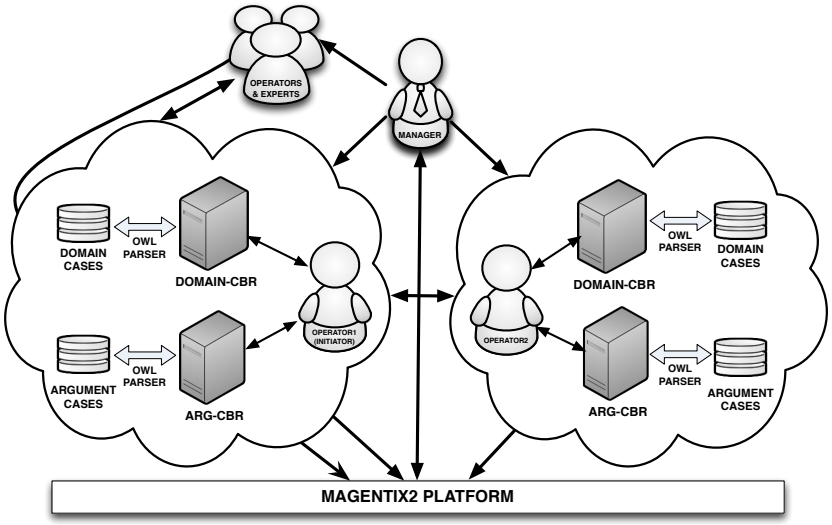

Figure 4: Architecture of the Call Centre Application

position is attacked but the opponent agent cannot defeat it, the position receives one vote. Otherwise, the proponent of the position must withdraw it with the locution *noCommit*.

6. The dialogue ends when an agreement is reached (i.e. only one position remains undefeated in the argumentation dialogue) or a deadline is met. The best solution is proposed to the user and feedback is provided and registered by each technician helpdesk. In this way, agents update their case-bases with the information of the actual solution applied and the arguments generated during the argumentation dialogue. Note that if there is no agreement reached, the best solution can be selected by using different policies, such as to select the most voted position or the most frequent.

The helpdesk of each technician is provided with a case-based reasoning engine that helps them to solve the ticket. The new argumentation module will allow different technicians to reach agreements over the best solution to apply in each specific situation. In this example application, we assume that the most efficient technicians are acknowledged and rewarded by the company. Therefore, each technician follows a *persuasion* dialogue with their partners, trying to convince them to accept its solution as the best way to solve the ticket received, while observing the common objective of providing the best solution for a ticket from its point of view. Note that with this approach, we are assuming that the CBR process is working properly and as the systems evolves, agents learn correct solutions in their domain-cases case base and useful arguments in their argument-cases case-base, hence providing more accurate solutions. The next sections present the experimental settings that we have used to run the tests and the results achieved.

4.1. Experimental Settings

For the tests, a real database of 200 tickets solved in the past is used as domain knowledge. Translating these tickets to domain-cases, we have obtained a tickets case-base with 48 cases (note that several tickets could report the same problem and thus, can be represented by the same domain-case). Despite the small size of this case-base, we have rather preferred to use actual data instead of a larger case-base with simulated data, which could not necessarily perform in the same manner in a real application, capture the uncertainty that agents face concerning the social context of their opponents or include losses of information. Thus, this case-base

is populated with heterogeneous cases which features can include omissions, typos, etc. The argument-cases case-bases of each agent are populated with argument-cases selected from a set, obtained previously by training the system in several rounds to allow agents to learn argument-cases.

To diminish the influence of random noise, for each round in each test, all results report the average and confidence interval of 48 simulation runs at a confidence level of 95%, thus using a different ticket of the tickets case-base as the problem to solve in each run. The results report the mean of the sampling distribution (the population mean) by using the formula:

$$\mu = \bar{x} \pm t * \frac{s}{\sqrt{n}} \tag{9}$$

where, $\bar{x}$ is the sample mean (the mean of the 48 experiments), t is a parameter that increases or decreases the standard error of the sample mean ($\frac{s}{\sqrt{n}}$), s is the sample standard deviation and n is the number of experiments. For small samples, say below 100, t follows the *Student's t-distribution*, which specifies a certain value for the t parameter to achieve a confidence level of 95% for different sizes of population. In our case, with a population of 48 experiments the Student's t-distribution establishes a value of 2.0106 for t.

In each simulation experiment, an agent is selected randomly as *initiator* of the discussion. This agent has the additional function of collecting data for analysis. However, from the argumentation perspective, its behaviour is exactly the same as the rest of agents and its positions and arguments do not have any preference over others (unless there is a dependency relation that states it). The initiator agent receives one problem to solve per run, selected randomly from the tickets case-base. Then, it contacts its partners (the agents of its group) to report to them the problem to solve. If the agents do not reach an agreement after a maximum time, the initiator chooses the most supported (the most voted) solution as the final decision (or the most frequent in case of draw). If the draw persists, the initiator makes a random choice among the most frequent solutions.

The main objective of the tests is to evaluate the performance of an agent with an argumentative profile and different combinations of the tactics proposed when it engages in argumentation dialogues with agents with other profiles. The argumentative profile presented in Section 3 exploits all knowledge resources of the case-based argumentation framework proposed. Other agent profiles involve different modifications of this profile. Therefore, to perform these tests, an agent (say the initiator agent) has been set to have an *argumentative* profile and to follow different tactics to argue with agents of different profiles. Also, these agents do not follow any specific tactic (assign the same weight to all parameters of the support factor to select positions and arguments, but still use the similarity degree and this support factor to perform this selection). In our evaluation scenario, we have assumed that all agents know each other and all are in the same group, responsible for providing support to problems reported in a specific project that the company is running. The influence of different degrees of friendship and group membership are difficult to evaluate with the limited amount of data of our tickets case-base and remains future work. By default, the dependency relation between technicians is *charity*. The values of each agent have been randomly assigned from a set of pre-defined values (*efficiency* of the problem solving process, *accuracy* of the solution provided and *savings* in the resources used to solve the ticket) and agents know the values of their partners. Also, if not specified otherwise, all agents play the role of *operator*.

First, we have tested the performance of the enhanced system in comparison with the original system, which proposes solutions by using the CBR methodology, but does not allow agents to argue about their propositions. The *performance tests* have been repeated and their results compared for the following decision policies:

- Random policy (CBR-R): each agent uses its domain CBR module to propose a solution for the problem to solve. Then, a random choice among all solutions proposed by the agents is made. This policy represents the operation of the original system, where agents do not have an argumentation CBR module and are not allowed to argue.

- Majority policy (CBR-M): each agent uses its domain CBR module to propose a solution for the problem to solve. Then, the system selects the most frequently proposed solution. This policy represents the operation of the original system, where agents do not have an argumentation CBR module and are not allowed to argue.

- Argumentation policy (CBR-ARG): agents have domain and argumentation CBR modules. Each agent uses its domain CBR module to propose a solution for the problem to solve and its argumentation CBR module to select the best positions and arguments to propose in view of its argumentation experience. Then, agents perform an argumentation dialogue to select the final solution to apply.

The ability of the framework to represent the social context of the system has also been evaluated. For the *social context tests*, an agent (say, the initiator) has been allowed to play the role of an *expert* or *manager*. An expert is an agent that has specific knowledge to solve certain types (categories) of problems and has its case-base of domain-cases populated with cases that solve them. Thus, the expert domain-cases case-base has as much knowledge as possible about the solution of past problems of the same type. That is, if the expert is configured to have 5 domain-cases in its domain-cases case-base, and there is enough suitable information in the original tickets case-base, these cases represent instances of the same type of problems. In the case that the tickets case-base has less than 5 cases representing such category of problems, for instance three cases that stand for "hardware errors", the remaining two cases are both of the same category (if possible), for instance "kernel errors". In our case, the expert agent has an *authorization* dependency relation over other technicians. Therefore, if it is able to propose a solution for the ticket requested, it can generate arguments that support its position and that will defeat other operators' arguments, due to its prevailing dependency relation. All simulation tests have been executed and their results compared for the random based decision policy (CBR-R Expert), the majority based decision policy (CBR-M Expert) and the argumentation based policy (CBR-ARG Expert). For these policies, the domain-cases case-base of the expert has been populated with expert domain knowledge. To evaluate the global effect of this expert knowledge, the results obtained for the accuracy of predictions when the domain-cases case-base of all agents are populated with random data are also shown for each policy (CBR-R, CBR-M and CBR-ARG).

A manager is an agent whose decisions have to be observed by other agents with an inferior role in the system, due to its higher position in the company hierarchy of roles. A manager can be a technician with specialised knowledge about the best solution to apply to specific types of problems, or else, a technician that has access to more information than an operator, due to its highest rank in the company. To simulate the presence of a manager in the system, we have considered the last case and populated the manager domain-cases case-base with twice the information that the domain-cases case-base of the operators (until a maximum amount of 45 domain-case). In our case, the manager agent has a *power* dependency relation over other technicians. Therefore, if it is able to propose a solution for the ticket requested, it can generate arguments that support its position and that will defeat other operators' arguments, due to its prevailing dependency relation. All simulation tests have been executed and their results compared for the random based decision policy (CBR-R Manager), the majority based decision policy (CBR-M Manager) and the argumentation based policy (CBR-ARG Manager).

Again, to evaluate the global effect of these different configurations of the domain knowledge, the results obtained for the accuracy of predictions when the domain-cases case-base of all agents are populated with random data are also shown for each policy (CBR-R, CBR-M and CBR-ARG).

Due to the small size of the whole tickets case-base, the performance and social context tests have been executed with a number of 7 agents participating in the dialogue, with domain-cases case-bases with a maximum number of 45 domain-cases. Thus, the domain-cases case-bases of the agents have been randomly populated and increased by 5 cases (from 5 to 45 cases) in each experimental round. Also, if agents have argument-cases case-bases, these have been populated with 20 randomly selected argument-cases (from the set obtained previously by training the system in several rounds to allow agents to learn argument-cases). For testing the performance of the argumentation policy and the ability of the framework to represent the social context of the agents, we have considered that all agents have an argumentative profile and follow a persuasive tactic, which selects such arguments that were more persuasive in the past. After that, the advantages of following this or any other tactic are evaluated in depth in the *strategy tests*.

Concretely, we have compared different dialogue strategies (as combinations of an agent profile and the tactics proposed in Section 3.2) that an agent can follow to argue with other agents. We consider the dialogue strategies defined in Section 3.2, which combine each tactic proposed with the argumentative profile presented in Section 3.1. Recalling, we have the Persuasive Strategy (ST1); the Maximise-Support Strategy (ST2); the Minimise-Risk Strategy (ST3); the Minimise-Attack Strategy (ST4); the Maximise-Efficiency Strategy (ST5); and the Explanatory Strategy (ST6). Also, we have compared these results with the performance of a random strategy (STR) for any type of agents involved in the dialogue. Following this strategy, an agent selects positions and arguments at random without any underlying criteria. These strategies are evaluated by computing the agreement percentage obtained by the agents when they have a *low* (from 5 to 15 domain-cases), *medium* (from 20 to 30 domain-cases) or *high* (from 35 to 45 domain-cases) knowledge about the domain. Also, the argumentative agent has a full case-base of 20 argument-cases. Then, the percentage of agents that an argumentative agent (the initiator, for instance) is able to persuade to reach an agreement to propose its solution as the best option to solve a ticket is computed for the same settings (from low to high knowledge about the domain). To be able to follow effectively an argumentation tactic, an agent needs to have useful argument-cases to reuse these experiences for the current argumentation situation. Thus, all strategic tests report results obtained when the initiator agent has some useful argument-cases that match the current situation and actually uses them to select the best position and arguments to propose.

Finally, to evaluate the effect of the available argumentative knowledge that agents have, we have run a test for the following specific settings of the argumentation policy. These settings cover the more interesting options regarding which agents have argumentation skills:

- CBR-ARG All-Argument-Cases (CBR-ARG AAC): All participating agents have argument-cases in their argument-cases case-base.

- CBR-ARG Initiator-Argument-Cases (CBR-ARG IAC): Only one agent, say the initiator agent, has argument-cases in its argument-cases case-base. Note that the selection of the initiator as the agent that has argumentative knowledge is just made for the sake of simplicity in the nomenclature. The behaviour of this agent only differs from the other agents' in the fact that it is in charge of starting the dialogue process and conveying the information about the final outcome. This does not affect its performance as dialogue participant and does not grant this agent any privileges over their partners.

- CBR-ARG Others-Argument-Cases (CBR-ARG OAC): All agents except from one, say the initiator, have argument-cases in their argument-cases case-bases.

In this test, if agents have argumentation knowledge, their argument-cases case-bases have been populated with different argument-cases from a set obtained previously by training the system in several rounds, allowing agents to acquire argumentation knowledge. The next section shows and analyses the results obtained.

4.2. Evaluation Results

4.2.1. Performance Tests

To evaluate whether the enhanced system actually improves the performance of the original system, the percentage of problems that the system is able to solve providing a correct solution is computed for each decision policy. To check the solution accuracy, the solution agreed by the agents for each ticket requested is compared with its actual solution, stored in the tickets case-base. Note that we have made the assumption that there is only one possible solution for a problem, being the one that is stored in the tickets case-base. However, with the system working in a real situation, an agent could come up with another solution that is completely different but does actually work. Nevertheless, this simplification allows us to test the system in a controlled environment without the need for having human experts evaluating if an alternative solution is also appropriate.

One can expect that with more knowledge stored in the case-bases, the number of problems that were correctly solved should increase. Figure 5 presents the results of the performance tests. These tests show that as the number of domain-cases of the agents' case-bases increases, the solution proposed by them is more appropriate and similar to the actual solution registered in the tickets case-base for the ticket that has been requested to the agents (the mean error percentage in the solution predicted decreases). Obviously, if agents have more information in their case-bases, the probability that one or more of them have a suitable domain-case that can be used to provide a solution for the current problem increases. This applies also in the case of the random policy, although this policy never achieves the 100% of correct solution predictions. Also, the results achieved by the argumentation policy improve those achieved by the other policies, even when the domain-cases case-bases are populated with a small number of cases. The argumentation policy achieves more than twice the solution accuracy than the other policies for a domain-cases case-base size up to 15 cases. These results demonstrate that if agents have the ability of arguing, the agents whose solutions are more supported by evidence have more possibilities of wining the argumentation dialogue and hence, the quality of the final solution selected among all potential solutions proposed by the agents increases. Therefore, the enhanced system, which allows the agents to engage in argumentation dialogues, actually outperforms the original system in the accuracy of the solutions provided.

4.2.2. Social Context Tests

In this test, an agent has been allowed to play the role of an *expert*, while the rest of agents play the role of *operators*. Figure 6(left) shows how the accuracy of predictions is higher if agents are allowed to argue following the CBR-ARG Expert policy. Comparing the results obtained when the initiator has (CBR-R Expert, CBR-M Expert and CBR-ARG Expert) or does not have expert knowledge (CBR-R, CBR-M and CBR-ARG), as expected, agents are able to reach better accuracy in their final prediction when they are able to argue and there is an expert participating in the argumentation dialogue (CBR-ARG Expert). This demonstrates that the decisions of the expert prevail and, as it has more specialised domain-knowledge to propose solutions, the predictions of the system are more accurate.

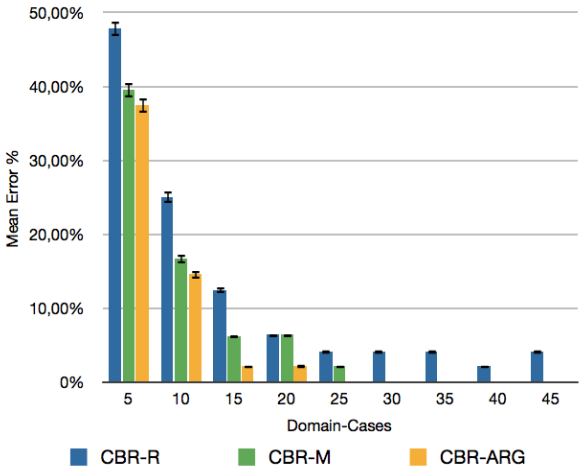

Figure 5: Solution prediction accuracy achieved by 7 operators ([5, 45]Δ5 domain-cases; 20 argument-cases)

In the following test, an agent has been allowed to play the role of a *manager*, another agent plays the role of *expert* (with the same configuration for its domain-cases case-base as in the previous test) and the other agents play the role of *operators*. Figure 6(right) shows that although the manager has more domain knowledge, as this in not specialised knowledge to solve any type of problems, its predictions can have less quality than the other agents' predictions. However, as in our case-based argumentation framework the decisions of managers have the highest priority due to its prevailing dependency relation over other roles (even when there is an expert participating in the dialogue), these low quality solutions are accepted as the final solutions that the system proposes. Therefore, until all agents have a sufficient amount of domain knowledge to provide accurate solutions (up to 20 domain-cases), the argumentative policy (CBR-ARG Manager) gets worse results in prediction accuracy than the other policies. In fact, when agents have low knowledge about the domain, as the positions of the manager are selected over the expert's positions, the systems gets a poor accuracy in its predictions, quite worse than in the case that all agents have the same amount of random knowledge about the domain.

4.2.3. Strategy Tests

Table 2 shows the percentage of times that an agreement about the best solution to apply is reached when the argumentative agent is arguing with agents that have an *agreeable* profile. Also, Table 3 shows the percentage of agreeable agents that the argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket. Agreeable agents do not challenge the positions of other agents, but only try to defend their positions if they are attacked. However, an agreeable agent accepts the position of another agent that has proposed its same position or a position that it has generated, although it has not ranked it as the best solution to apply. That means that in these cases, the agreeable agent votes on the position of other agents and withdraws its own.

For low knowledge about the domain, few (if any) useful argument-cases are found and agents cannot reach an agreement about the best solution to apply to solve the ticket at hand, whatever strategy is followed. An useful argument-case is a case that matches the current problem description (both domain and social contexts) and proposed the same solution as the one whose suitability is being evaluated in the current situation. This does not mean that

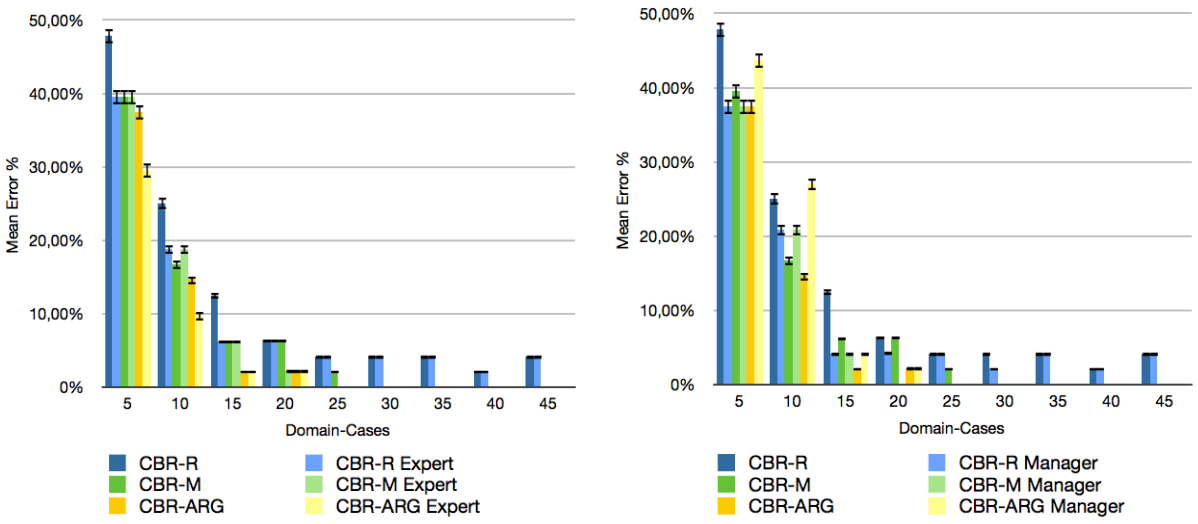

Figure 6: Solution prediction accuracy achieved by: (left) 1 expert and 6 operators; (right) 1 manager, 1 expert and 5 operators ([5, 45]Δ5 domain-cases; 20 argument-cases)

the system is not able to propose a solution at all, but the solution proposed is not agreed by all agents. In the case of having a medium amount of knowledge about the domain, the best results are achieved for these strategies that minimise the probability or the number of potential attacks that an argument can receive (ST3 and ST4). Remember that an agreeable agent does not attack any position and if attacked, if it cannot defend itself, it just withdraws its position. Therefore, if the initiator uses arguments to generate its position and arguments that the agreeable agent cannot defeat, the initiator will have less agreeable agents competing in the dialogues and an agreement is reached more easily. However, if the agents have more knowledge about the domain, the agreeable agents increase their options to generate potential attacks to rebut the attacks that they receive, and hence, following a strategy that selects those positions and arguments that are expected to have a higher acceptability degree from the other agents (ST1) makes the initiator to be able to reach most agreements.

The results of Table 3 show again how ST3 and ST4 perform better for a medium amount of knowledge about the domain and ST1 for a high amount of knowledge. As pointed out before, these strategies minimise the probability or the number of potential attacks that an argument can receive. Therefore, they make the initiator agent to reach to a higher number of agreements by persuading other agents to accept its position as the best solution to apply for a ticket requested. Moreover, results show how the random strategy is outperformed by any other strategies both in agreement percentage and percentage of agents persuaded for any amount of knowledge about the domain.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**	**STR**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	25.00%	16.67%	*33.33%*	*33.33%*	16.67%	16.67%	0.00%
HIGH	*63.89%*	55.56%	50.00%	25.00%	38.89%	19.44%	13.10%

Table 2: Agreement Percentage with Agreeable agents.

Strategy / Domain Knowledge	ST1	ST2	ST3	ST4	ST5	ST6	STR
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	14.29%	11.90%	*22.62%*	*22.62%*	11.90%	11.90%	0.00%
HIGH	*52.38%*	45.24%	38.89%	21.03%	30.95%	14.68%	11.22%

Table 3: Percentage of Agreeable agents persuaded.

Table 4 shows the percentage of times that an agreement about the best solution to apply is reached when the argumentative agent is arguing with agents that have a *disagreeable* profile. In addition, Table 5 shows the percentage of agents that the argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket when it is arguing with disagreeable agents. Disagreeable agents act similarly to agreeable agents, do not challenge any position proposed by other agents, but in this case, this profile of agents only accept the position of a partner (voting it and withdrawing its own), if it exactly coincides with the position that they have proposed as the best solution to apply.

Again, low knowledge about the domain prevents the use of useful argument-cases. However, for both medium and high amounts of knowledge about the domain, the initiator agent is able to convince more agents if it follows a strategy that minimises the number of potential attacks that its position and arguments can receive, getting thus to higher agreement percentages. This results in selecting those positions that, although still serve the initiator agent's objectives, are more similar to the positions proposed by the disagreeable agents and hence, these have less potential attacks to put forward and are more easily persuaded to reach an agreement to accept the initiator's position (as shown in Table 5). Nevertheless, disagreeable agents are difficult to convince and both agreements and agents that agree percentages are low independently of the strategy followed by the initiator. In fact, results show how the random strategy is not able to reach any agreement for any amount of knowledge about the domain.

Strategy / Domain Knowledge	ST1	ST2	ST3	ST4	ST5	ST6	STR
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	16.67%	16.67%	8.33%	*25.00%*	8.33%	16.67%	0.00%
HIGH	36.11%	30.56%	11.11%	*38.89%*	30.56%	25.00%	0.00%

Table 4: Agreement Percentage with Disagreeable agents.

Strategy / Domain Knowledge	ST1	ST2	ST3	ST4	ST5	ST6	STR
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	11.90%	11.90%	5.95%	*15.48%*	5.95%	11.90%	0.00%
HIGH	30.56%	25.40%	9.52%	*31.35%*	25.79%	23.02%	0.00%

Table 5: Percentage of Disagreeable agents persuaded.

Table 6 shows the percentage of times that an agreement about the best solution to apply is reached when the argumentative agent is arguing with agents that have an *elephant's child* profile. In addition, Table 7 shows the percentage of elephant's child agents that the argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket. Elephant's child agents have a weird profile that challenges any position proposed by

other agents, whatever this position is. These agents are used to overload the system with useless interactions between the agents. A side effect of this overload is that in most cases elephant-child agents are attacking positions that they themselves have generated and hence, they cannot find in their argument-cases case-bases useful cases to select the best attack arguments (since they have not actual arguments to attack them). Therefore, as our case-based tactics rely on these argument-cases to compute the support factor, the weights assigned to each parameter of the support factor have no effect and all underlying argumentation strategies get the same results.

An elephant's child agent only accepts the position of another agent if it challenges this position and the other agent wins the debate. This has more probability of occurring when agents have less knowledge about the domain and thus, less knowledge to generate positions and arguments. Therefore, whatever strategy the initiator follows, the percentage of agreement only exceeds 50% if agents have low knowledge about the domain and the initiator is able to persuade the 28.57% of agents, as shown in the tables, and decreases as the amount of domain knowledge increases. However, as pointed out before, in most cases elephant's child agents are attacking positions that they are able to generate and support and thus, if the attacked agents are defeated and withdraw their positions, this prevents the effective development of agreement processes, especially when agents have more domain knowledge and are able to propose more solutions. Similarly, the random strategy achieves as bad results as the other strategies with low and high knowledge about the domain and even worst with medium knowledge about the domain.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**	**STR**
LOW	66.67%	66.67%	66.67%	66.67%	66.67%	66.67%	66.67%
MEDIUM	16.67%	16.67%	16.67%	16.67%	16.67%	16.67%	8.33%
HIGH	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Table 6: Agreement Percentage with Elephant's Child agents.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**	**STR**
LOW	28.57%	28.57%	28.57%	28.57%	28.57%	28.57%	28.57%
MEDIUM	15.48%	15.48%	15.48%	15.48%	15.48%	15.48%	8.33%
HIGH	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%

Table 7: Percentage of Elephant's Child agents persuaded.

Table 8 shows the percentage of times that an agreement about the best solution to apply is reached when the argumentative agent is arguing with agents that have an *open-minded* profile. Also, Table 9 shows the percentage of open-minded agents that the argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket. An open-minded agent only challenges the positions of other agents if it has not been able to generate such positions (they are not its proposed position or a position in its list of potential positions to propose). However, what specially distinguishes the behaviour of this agent from other agent profiles is the fact that it accepts the position of an agent that has started and won a dialogue with it, although this position is not in its list of potential positions.

As for agreeable and disagreeable agents, if the knowledge that the agents have about the domain is low, no agreement is reached. For the case of having a medium amount of domain knowledge, the agreement is reached easily if the initiator follows a strategy that selects the most

potentially persuasive arguments or those that cover as much justification elements as possible (ST1 and ST6). If the initiator is able to better defend its position with more persuasive arguments or more support elements, it ensures that its position will prevail accepted and thus, the defeated open-minded agents will withdraw their positions and agree to propose the initiator's as the best solution to apply. Thus, Table 9 shows a higher percentage of agents persuaded if the initiator follows ST1 and ST6. However, if agents have high knowledge about the domain, the percentage of agreements and agents persuaded are clearly higher when the initiator follows ST2 and ST6. This demonstrates that when the initiator agent has more knowledge and hence, more support degree (higher probability of being accepted at the end of the dialogue) and more elements to justify its positions and arguments, open-minded agents easily accept the position of the initiator and withdraw theirs when they attack the initiator's position. Finally, in almost all cases the random strategy is outperformed by the other strategies, except for the case of having medium amount of knowledge about the domain, where ST5 performance is particularly bad.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**	**STR**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	*41.67%*	33.33%	25.00%	33.33%	16.67%	*41.67%*	25.00%
HIGH	47.78%	61.11%	47.78%	52.22%	50.00%	*62.78%*	16.67%

Table 8: Agreement Percentage with Open-Minded agents.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**	**STR**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	*28.57%*	23.81%	16.67%	23.81%	11.90%	27.38%	13.10%
HIGH	37.46%	48.57%	37.86%	44.21%	38.89%	*51.19%*	15.48%

Table 9: Percentage of Open-Minded agents persuaded.

Table 10 shows the percentage of times that an agreement about the best solution to apply is reached when the argumentative agent is arguing with agents that also have an *argumentative* profile. Also, Table 11 shows the percentage of agents with its same profile that an argumentative agent (the initiator) is able to persuade to propose its solution as the best option to solve a ticket. An argumentative agent only challenges positions of other agents when they have proposed a position different from its position. In addition, this agent profile accepts such positions that it attacks when the opponent wins the confrontation.

Again, as in most strategy tests, low knowledge about the domain results in simple dialogues that do not reach any agreement and the solution proposed is not agreed by all agents. For a medium amount of knowledge about the domain, the percentage of agreements reached is higher when the initiator follows a strategy that minimises the probability that their positions and arguments are attacked (ST3). In fact, if positions are not attacked at all, they prevail as potential candidates to be selected as the final solution for the ticket requested. Note that in this test, all agents accept positions and arguments of other agents under the same circumstances, so decreasing the probability of a position to be attacked, increases its probability of being accepted at the end of the dialogue.

If argumentative agents have a high knowledge about the domain, many of them are able to propose accurate solutions and defend them against attacks. Therefore, such strategies that

allow the initiator to better defend its positions and arguments by increasing the probability of being accepted (ST2) or by reducing the number of potential attacks get better agreement percentages (ST4). If we understand that more acceptable and less attackable arguments lead to shorter dialogues, the good results achieved when the initiator follows a strategy that selects those positions and arguments that produced shorter dialogues in the past (ST5) can be viewed as a logical consequence of the good performance of ST2 and ST4. Table 11 shows a slightly higher percentage of persuaded agents when the initiator has deep knowledge about the domain and the number of potential attacks is decreased. As less attacks are received, the less effort the initiator needs to convince other agents that attack its position to agree and accept it as the best solution to apply for the requested ticket. Nevertheless, all ST2, ST4 and ST5 strategies get good percentages of persuaded agents for this amount of domain knowledge. Moreover, in these tests the random strategy achieves particularly bad results for any amount of domain knowledge.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**	**STR**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	38.89%	47.22%	*50.00%*	33.33%	33.33%	41.67%	16.67%
HIGH	82.22%	*100.00%*	82.22%	*100.00%*	*100.00%*	86.11%	16.67%

Table 10: Agreement Percentage with Argumentative agents.

Strategy / Domain Knowledge	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**	**STR**
LOW	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
MEDIUM	22.22%	29.76%	*30.95%*	21.43%	23.81%	29.76%	11.90%
HIGH	68.57%	80.56%	66.98%	*81.35%*	80.56%	71.43%	15.48%

Table 11: Percentage of Argumentative agents persuaded.

Summarising, Table 12 shows, for each strategy, each profile of the opponent agents and any amount of knowledge about the domain, the average of the percentage of agreement that the argumentative agent is able to achieve when useful argument-cases are used. Similarly, Table 13 shows the average of the percentage of agents that the argumentative agent is able to persuade when useful argument-cases are used. For strategies 1 to 6, the best results in the number of agreements reached and agents persuaded are obtained when all agents are argumentative. Among them, the strategy that guides the agent to propose positions and arguments that have a higher potential probability of being accepted (ST2) is the winning strategy for our customer support domain, followed closely by the strategy that makes the agent to minimise the number of potential attacks that its positions and arguments can receive (ST4). It seems reasonable that if we are measuring the percentage of agreements reached and the percentage of agents persuaded to reach them, the strategy that increases the acceptance probability of positions and arguments performs better. Also, if agents receive a lower number of attacks, it can be understood as a consequence of proposing positions and arguments that suit the objectives of more agents. The results of these experiments also show that the random strategy (STR) is significantly outperformed by the rest of strategies with all types of agents participating in the dialogue. Nevertheless, this strategy gets its best results when the initiator is arguing with open-minded agents. As pointed out before, this type of agents accept the position of an agent that has started and won a dialogue with them, although this position is not in their list of potential

positions. Therefore, when the initiator is selecting positions and arguments at random, they get better percentages of agreement and agents persuaded than the other types of agents.

Generally speaking, we can say that the use of argumentative agents improves the outcome of the agreement process. For the rest of the profiles, agreeable agents are more easily persuaded with positions and arguments that have a low probability of being attacked (probably because agreeable agents are also able to generate them); disagreeable agents are more easily persuaded with positions and arguments that will potentially get less number of attacks (probably because they are the same that disagreeable agents have proposed); and open-minded agents are easily persuaded with positions and arguments that have more elements that justify them (and hence the initiator has more elements to rebut attacks and win the dialogue). As pointed out before, elephant's child agents show a weird behaviour that gets low percentages of agreement and agents persuaded no matter which strategy the initiator follows. However, in most cases elephant's child agents are attacking positions that they are able to generate and support and thus, the attacked agents are defeated and withdraw their positions, which prevents the development of an effective agreement process.

Agent Profile \ Strategy	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**	**STR**
Agreeable	*29.63%*	24.07%	27.78%	19.44%	12.96%	12.04%	4.37%
Disagreeable	17.59%	10.19%	6.48%	*21.30%*	12.96%	13.89%	0.00%
Elephant's Child	5.56%	5.56%	5.56%	5.56%	5.56%	5.56%	2.78%
Open-Minded	29.81%	32.04%	24.26%	28.52%	22.22%	*34.26%*	13.89%
Argumentative	**40.37%**	**49.07%**	**44.07%**	**44.44%**	**44.44%**	**42.59%**	11.11%

Table 12: Average agreement percentage for all agent profiles.

Agent Profile \ Strategy	**ST1**	**ST2**	**ST3**	**ST4**	**ST5**	**ST6**	**STR**
Agreeable	*22.22%*	19.05%	20.50%	14.55%	10.32%	8.86%	3.74%
Disagreeable	14.15%	8.47%	5.16%	*15.61%*	10.58%	11.64%	0.00%
Elephant's Child	5.16%	5.16%	5.16%	5.16%	5.16%	5.16%	2.78%
Open-Minded	22.01%	24.13%	18.17%	22.67%	16.93%	*26.19%*	9.52%
Argumentative	**30.26%**	**36.77%**	**32.65%**	**34.26%**	**34.79%**	**33.73%**	9.13%

Table 13: Average percentage of agents persuaded for all agent profiles.

Finally, to evaluate the influence of the amount of argumentative knowledge of the agents on the agreement percentage, Figure 7 shows the results obtained by the argumentation policy when the number of argument-cases available for one or more agents is increased. In this test, all agents have an argumentative profile and follow a random dialogue strategy. When the initiator agent is the only agent that uses argumentative knowledge, as this knowledge increases, the probability of finding useful argument-cases to apply in each argumentation dialogue also increases. Therefore, this agent improves its argumentation skills and it is able to persuade the others to reach an agreement and accept its position as the best solution to apply for the ticket to solve. However, when almost all agents have a fair amount of argumentative knowledge, they all have acquired good argumentation skills and it is difficult to persuade an agent to accept the position of another agent. In fact, for the contents of the case-bases that we have used in this test, no agreement is reached in this case (CBR-ARG AAC with more than 6 argument-cases and CBR-ARG OAC with more than 2 argument-cases). Also, when almost all agents have a small quantity of argument-cases, the probability of finding a useful argument-case is very low.

In these cases (CBR-ARG AAC with 6 argument-cases and CBR-ARG OAC with 2 argument-cases), the performance of the system suffers from a high randomness, and this agent that finds a useful argument-case has a higher advantage over the others, being able to persuade them to reach an agreement that favours its preferences. Thus, the CBR-ARG IAC policy gets higher percentage of agreement when useful argument-cases are actually used.

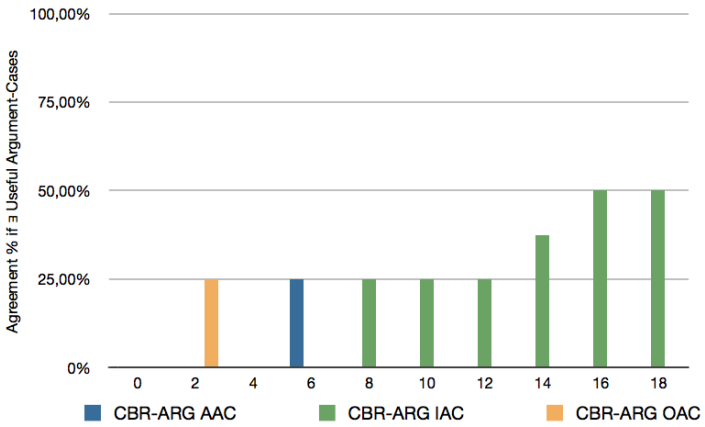

Figure 7: Percentage of agreement reached by 7 agents when useful argument-cases are available (20 domain-cases; [0, 18]Δ2 argument-cases)

4.2.4. Discussion

This section shows how the argumentative agents outperform other agent profiles and are able to persuade them and reach a higher percentage of agreements. However, allowing all agents to argue could entail extra communication costs. An analysis of these communication costs is out of the scope of this paper and can be found in [19, Chapter 6]. Summarising, we can say that the communication costs of arguing are reasonable and the number of interchanged locutions tends to stabilize when the percentage of agreements reached approaches to 100%, which makes the argumentative dialogue to be worthy despite of these extra costs.

On the one hand, the performance of the enhanced system has been evaluated and compared with the results achieved by the original system. The tests show how those agents that follow an argumentation policy are able to provide more accurate solutions to the problems that the system receives. Therefore, in the enhanced system the ability of the agents to argue allows those who have better arguments to support their decisions to win the argumentation dialogue. In this way, the solutions with higher quality are selected among those proposed.

On the other hand, the influence of the dependency relations among agents in the accuracy of the solutions that they agree upon has also been evaluated. Therefore, if an expert actually is better informed to assign better solutions to specific types of problems, the performance of the system improves. However, if a manager just has a larger amount of information than the operators, but this information does not make the manager propose more accurate solutions, the performance of the system is adversely affected.

Finally, the strategical tests performed evaluate the percentage of agreements reached and the percentage of agents of different profiles persuaded by an argumentative agent. The results obtained demonstrate that if all participants of the argumentation process have an argumentative profile, more agreements are reached and more agents support the final solutions provided by the system. For all strategies, the best results are obtained when all agents are argumentative.

Among them, the strategies that guide the agent that follows them to propose positions and arguments that have a higher potential probability of being accepted (ST2) and that minimise the number of potential attacks that an agent can receive (ST4) are the winning strategies.

The influence of the amount of argumentation knowledge that argumentative agents have in the number of agreements reached by the agents has also been evaluated. If only one agent has argumentation knowledge, as this knowledge increases, the amount of agreements reached also increases. This demonstrates that this agent is effectively using its argumentation knowledge to select the best positions and arguments to put forward in a dialogue with other agents. Thus, as many argumentation-related information an agent gathers from its partners in a dialogue, as more proficient the agent is to persuade them to accept its proposals. Recall that agents store in argument-cases information about the social context of their opponents and their group, as well as information about how persuasive arguments were for this specific domain and social context. However, if all or most agents have the ability of learning from argumentation dialogues, all of them have the same (high) persuasive power to defend their decisions and the agreement is difficult to achieve.

5. Related Work

Until recently, material about dialogue strategies in argumentation was hardly found [37], but in the last five years this area has experienced a growing interest. The theoretic research performed to date in the area of argumentation in MAS follows two differentiated approaches: the *heuristic* and the *game-theoretic*. However, there is no consensus on the definition of a strategy and on the parameters necessary for its definition [3]. Consequently, there is no standard methodology for the definition of dialogue strategies. A first attempt to model the process of strategy construction for negotiation dialogues was published in [39]. This work proposes a methodology for designing heuristic negotiation strategies that guides the strategy designer along several stages to produce modular specifications of tactics and strategies.

The literature on strategies for argumentation provides different definitions for the notion of strategy. Several examples are:

> "The strategy is a decision problem in which an agent tries to choose among different alternatives the best option, which according to its beliefs, will satisfy at least its most important goals [3]."

> "Private strategies, as adopted by an individual agent, specify the dialogue move(s) the agent is willing to utter, according to its own objectives and other personal characteristics.[24]"

> "A strategy of an agent specifies a complete plan that describes what action the agent takes for every decision that a player might be called upon to take, for every piece of information that the player might have at each time that it is called upon to act. Thus a strategy for an agent would specify for each possible subset of arguments that could define its type (the set of arguments that the agent is capable of putting forward), what set of arguments to reveal.[35, Chapter 16]"

Our notion of strategy takes into account the profile of the agent and the tactic that it is following (which in its turn, depends on the contents of its knowledge resources) and thus, we endorse the definition of [24].

There are different approaches to the study of strategies in argumentation frameworks and in dialogical systems in general. On one hand, preliminary works studied the concept of strategy

as developing heuristics for move selection in argumentation dialogues. A first contribution was provided in [13]. In this work the author defines a *Toulmin dialogue game machine* and proposes some heuristics for move selection. The acceptability of the arguments is computed by using some Toulmin-like rules. A similar work is the one presented in [5], which proposes heuristics for move selection on the context of persuasion and negotiation dialogues. This research uses the three-level approach of strategy proposed in [28]. The levels identified are:

1. maintaining the focus of the dispute.
2. building one's point of view or destroying the opponent's one.
3. selecting the method to fulfil the objective set at levels 1 and 2.

While levels 1 and 2 refer to *strategy* (planning the line of argumentation), level 3 refers to *tactic* (the means to reach the aims fixed at the strategical level). Then, the account for strategy proposed follows three steps to develop strategies:

1. define some agent profile: agreeable (accept whenever possible), disagreeable (only accept when there is no reason not to), open-minded (only challenge when necessary), argumentative (challenge whenever possible), or elephant child (question whenever possible).
2. choose to build or destroy.
3. choose some appropriate argumentative content.

Opposite to [13], [5] computes argument acceptability by using a more general Dung-like argumentation framework. In a subsequent work, the author studied the notion of strategy for selecting offers during a negotiation dialogue [4], proposing different agent's profiles and different criteria for the notions of acceptability and satisfiability of offers. Also, [3] views dialogue strategies as a two step decision process: i) to select the type of act to utter at a given step of a dialogue, and ii) to select the content which will accompany the act. Thus, an agent tries to choose among different alternatives the best option, which according to its beliefs, will satisfy at least its most important goals. There are two types of goals: *strategic goals*, which help an agent, on the basis of the strategic beliefs, to select the type of act to utter; and *functional goals*, which help an agent to select, on the basis of the basic beliefs, the content of a move. Then, the work proposes a formal model for defining strategies. The model takes as input the strategic and the functional goals together with the strategic and basic beliefs and returns the next move (act plus its content) to play. Then, the model assesses each alternative by constructing the set of supporting arguments for each one and evaluating their quality.

The agent profiles of [5] were also considered in [24] to develop different types of strategies. This work proposes an argument-based framework for representing communication theories of agents that can take into account the conformance to society protocols, private tactics of individual agents, strategies that reflect different types of personal attitudes (agents' profiles) and adaptability to the particular external circumstances at the time when the communication takes place. Although the authors do not provide a clear structure and definition for their notion of agent society, social relations between agents are captured in the form of preference rules that affect the tactic component of an agent and help it to decide the next move in a dialogue.

The work presented in [28] has also inspired subsequent research on strategies for human-computer dialogue and educational debate. Recently, in [50] a set of strategic heuristics for a human computer debating system regulated by the dialogue game DE (an amended dialogue model based on the dialectical system DC [27]) have been proposed. This work presents strategic heuristics for a computer to enable it to function as a dialogue participant. The proposed strategies define which is the best move a dialogue participant can make, based on the previous move made by the opponent. Opposite to our case-based strategies, which are built on the agents' profile and tactic (the latter based on its turn on the knowledge that agents gained

from previous dialogues), these heuristics heavily rely on judgements of quality by their author. Also, this approach has been developed and tested in a situation where there are two agents interacting, both following the same strategy or one of them following a strategy and the other acting at random, and both sharing the same knowledge base or a subset. Thus, it quite differs from our open MAS scenario, where there are several agents with completely different knowledge bases that can argue following different argumentation strategies. Also in the line of [28], the work presented in [51] proposed a probabilistic-based approach for selecting the move with a highest probability of winning a computer game for abstract argumentation.

Another alternative approach for the development of heuristic dialogue strategies looks at how an agent may decide what utterance to advance by making use of *opponent modelling*. This is the case of the work presented in [31], which examines possible utterances in each step of the dialogue, creating a tree of possible dialogues and assigning them an *utility value*. This utility may depend on many factors, including the arguments introduced within the dialogue, the agent introducing these arguments or the admissible arguments under some semantics. The opponent model includes a knowledge base with the arguments that the opponent can play and its utility function. Therefore, this approach assumes that the proponent of an argument has at least some partial knowledge about the arguments that the opponent can put forward and the utility that it assigns to them. Also, it uses a Dungian argumentation framework [18], which abstracts the actual structure of arguments and the attack relation among them. A similar approach was taken in [15] to model agents' beliefs in abstract argumentation frameworks. However, the application of these proposals in open MAS scenarios is not straightforward. In open MAS, heterogeneous agents can enter or leave the system continuously and in principle, agents do not know the set of arguments that their opponents can play. Therefore, this would imply a thorough effort to learn (possibly many) opponent models during the dialogue. To some extent, our approach for saving as cases the arguments that specific agents have interchanged in previous dialogues and using later this information to select the best argument to put forward in a similar situation applies a lazy learning method to model the expected behaviour of opponents.

Further work in modelling opponent behaviour to devise argumentation strategies has been recently published in [14]. This work proposes a novel approach to learn an opponent *value model* that allows agents to select the appropriate arguments in order to come to an agreement on how to act in a deliberation dialogue. The deliberation strategy presented allows a proponent to use its perception of the recipient to guide its dialogue behaviour and make proposals that are more likely to be agreeable. Thus, in each step of the dialogue, among the set of potential arguments to advance, an agent selects such argument that promotes its preferences and also the preferences of its opponent, from its point of view. In addition, agents dynamically modify their value model of their opponent as the dialogue proceeds. As in our work, this proposal takes into account the values that arguments promote to devise an argumentation strategy. Concretely, it adapts a previous action based alternating transition system that uses practical reasoning as presumptive argumentation [9]. However, opposite to us, the authors have assumed that there are only two participants in the dialogue and that each is following the same strategy. Also, the dependency relations between agents are not considered in the argument management process.

On the other hand, a different approach follows a game-theoretic view to the study of dialogue strategies. This is the case of the work proposed in [43], where the probability of a conclusion is calculated using a standard variant of defeasible logic, in combination with standard probability calculus. In this approach the exchange of arguments is analysed with game-theoretic tools, yielding a prescriptive account of the actual course of play. Other game-theoretic approach for the study or dialogue strategies in negotiation dialogues was presented in [35]. This approach uses the paradigm of *Argumentation Mechanism Design (ArgMD)* for designing and analysing argument evaluation criteria among self-interested agents using game-theoretic tech-

niques. Mechanism design (MD) is a sub-field of game theory concerned with determining the game rules that guarantee a desirable social outcome when each self-interested agent selects the best strategy for itself. The approach analyses strategy-proofness under grounded semantics for a specific type of arguments, the so-called *focal arguments* (the arguments that agents are especially interested in being accepted). In further research [36] the ArgMD approach has been applied to more realistic situations in which each agent has a single focal argument it wishes to have accepted. Opposite to the heuristic-based approaches, the goal of this game-theoretic approach is to design rules that ensure, under precise conditions, that agents have no incentive to manipulate the outcome of the game by hiding arguments or lying (how to ensure the truth in an argumentation framework).

As pointed out in Section 3, we are closer to the heuristic approach of [3] rather than to the game theoretic ones. One one hand, game theoretic approaches are usually applied to abstract argumentation frameworks where the strategies of agents determine which argument(s) they will reveal in each argumentation step. Common objectives of these works are to study the conditions under which the outcome of the game is not affected by the strategic decisions of the agents or to predict the outcome of the game. In contrast, we define dialogue strategies on basis of the knowledge that agents have about the domain, previous argumentation experiences and the social context (the roles, preferences over values and dependency relations among agents and groups). In doing so, we take into account the specific structure of arguments and the knowledge resources of our framework and design strategies to help agents to take advantage over other participants in the argumentation dialogue.

On the other hand, in our application domain there is not a pre-defined utility function about the payoff that an agent gets for the fact of winning the dialogue or having accepted more or less arguments, which is one of the common assumptions in game theoretic approaches for strategic argumentation. Despite that, in the customer support application developed, we have assumed that the most efficient technicians could be acknowledged by the company. Therefore, each technician follows a *persuasion* dialogue with their partners, trying to convince them to accept its solution as the best way to solve the ticket received, while observing the common objective of providing the best solution for a ticket from its point of view. However, this implicit payoff is neither computed by the system, nor used to guide the dialogue strategy. Finally, game theory assumes complete knowledge of the space of arguments proposed in the argumentation framework. This assumption is unrealistic in an argumentation dialogue between heterogeneous agents which have individual and private knowledge resources to generate arguments.

Further disadvantages of applying game theory to devise dialogue strategies in argumentation frameworks are similar as those reported in [23] for the problem of applying game theory to automated negotiation. First, game theoretic studies of rational choice in multi-agent encounters typically assume that agents are allowed to select the best strategy from the space of all possible strategies, by considering all possible interactions. This is computationally intractable when the space of possible choices grows. Also, game theory assumes that it is possible to characterise an agent's preferences about possible outcomes. This is hard to define for agents that represent humans that are engaged in an argumentation process (e.g. privacy considerations, humans can be dishonest and self-interested when revealing their intentions, etc.). Our alternative solution is to define preferences over values instead of preferences over dialogue outcomes and use them to guide the agents' choices.

As pointed out in Section 1, many argumentation systems produce arguments by applying a set of inference rules [13][5][24][26], which require to elicit an explicit model of the domain. In the context of dynamic open MAS, this model is difficult to specify in advance. However, storing in cases the arguments that agents put forward and reuse later this information could be relatively simple. The work done in the eighties and nineties about legal CBR fostered the

argumentation research in the AI community. The first and probably the most important case-based argumentation system of that time was the HYPO system [8]. HYPO generates legal arguments by citing previous cases (precedents) as justifications of the conclusions about who should win a dispute in the domain of the American trade secrets law. Concretely, the system generates *3-ply arguments* that show the best cases to cite for each party, their distinctions (the differences between the current and the previous case) and their counterexamples (previous cases that are similar to the current case and that were resolved in the other party's favour). HYPO's success gave rise to the subsequent development of several systems that share its problem analysing style. The intelligent tutoring system CATO [2], which teaches law students to build arguments from cases; the system BankXX [42], which generates arguments in the domain of American bankruptcy law; and the CABARET system [41], which produces legal arguments in the domain of the American tax law are some examples. HYPO and its descendants are *retrospective* systems, in the sense that they look back to reconstruct or explain what happened in an argumentation dialogue. By contrast, our case-based argumentation framework is *prospective* and intended for modelling systems that do not exist yet by using the MAS paradigm. Thus, in our framework previous argumentation experiences stored in the form of cases are not used to generate explanations about the past, but new arguments and positions to deal with current problems to solve.

Nowadays, the argumentation research in CBR continues being very active and, in fact, some approaches that integrate CBR in MAS to help argumentation processes have already been proposed [20]. However, the contributions are still scarce and much research in the area remains to be done. From our point of view, an interesting role that the CBR methodology can play in argumentation processes in MAS is to generate heuristic dialogue strategies based on previously acquired experience. Note that one of the main advantages of using CBR to manage argumentation in MAS is that it allows agents to learn from the process. Some of the few case-based argumentation frameworks found in the literature partially store the information about the current argumentation dialogue in the form of cases when the process finishes [25, 49]. In addition, the agents of the *AMAL* case-based argumentation framework presented in [29] can also learn during the argumentation dialogue by storing in their case-bases the cases that they receive from other agents. However, by contrast with our proposal, they do not learn how to predict the expected acceptability of arguments, but only increase their own knowledge with the knowledge that other agents share with them.

As shown in this paper, CBR can also be used to improve agents' argumentation skills and generate arguments to perform strategic argumentation that would easily persuade agents with similar social contexts. Some preliminary steps in this way have already been taken in related work that tries to learn agent profiles (again, modelling opponents) by using CBR. The first attempt to use CBR to provide information for building agents' profiles was performed in the *PERSUADER* system [48]. In this framework a mediator agent uses the information about previous negotiation processes stored in the case-base to develop the behavioural model of an unknown agent and devise the best way to persuade it. Similarly, in the negotiation framework proposed in [47] the information of the cases is used to decide which type of arguments are best suited to convince an agent with a specific profile and to infer other parameters that influence the negotiation process (e.g. time constraints, resources usage, etc.). Nevertheless, in both cases the dialogue strategy is highly domain dependent and completely relies on previous knowledge. Although in PERSUADER the agents' models can be dynamically updated, the preference order that determines which argument must be ultimately posed depends on a pre-established hierarchy. In our approach, strategies are developed taking into account the agent's profile and tactic, based on the elements of the argument-cases. As argument-cases have a generic structure, our case-based argumentation strategies can be applied in different domains where

the argumentation framework is implemented.

In a more dynamic and online way, the case-base could be used to store information about the agents' profile that could be gathered either by observing the current agents' behaviour, by learning information that the agents send during the dialogue or as a result of inquiry and information seeking processes. Therefore, this information could be used in the current argumentation process to generate and select better arguments to put forward and to evaluate the incoming ones. Case-based dialogue strategies have to be further investigated and there is still much work to do in this area.

6. Conclusions

In this paper we have presented different dialogue strategies (as combinations of an agent profile and different tactics) that an agent can follow to argue with other agents in a case-based argumentation framework for agent societies. To evaluate this proposal, the framework has been implemented in a real customer support application. However, the framework is generic enough to be applied in any domain where several agents are arguing about the best solution for a problem characterised as a set of features. Thus, in [21] our case-based argumentation framework was applied in a water market domain, where a set of agents argue about the best allocation for water resources. Future work will evaluate further combinations of profile-tactic for the initiator and also for the rest of the agents participating in the dialogue. Also, future evaluation tests will analyse the behaviour of the system when a mix of agents with different profiles participate in the argumentation dialogue.

For simplicity purposes, we have assumed in this work that a proponent agent addresses its arguments to an opponent of its same group, having complete knowledge of the opponents' social context. However, in real systems, some features of argument-cases could be unknown. For instance, the proponent of an argument obviously knows its value preferences, probably knows the preferences of its group but, in a real open MAS, it is unlikely to know the opponent's value preferences. However, the proponent could know the value preferences of the opponent's group or have some previous knowledge about the value preferences of similar agents playing the same role as the opponent. If agents belong to different groups, the group features could be unknown, but the proponent could use its experience with other agents of the opponent's group and infer them. Therefore, many interesting questions on how to infer the opponents' social context remain future work. A battery of tests to evaluate the influence in the performance of the system of the knowledge that an agent has about the social context of its opponents was developed and analysed in [19, Chapter 6].

We have assumed in our example that agents do their best to win the argumentation dialogue, thus following a persuasion dialogue, since in this way they can get economical rewards and increase prestige. Despite that, those solutions that are better supported prevail. Hence, if agents do not follow a dialogue strategy that leads the final outcome of the dialogue to fit their individual objectives, no matter if they give rise to the best solution to apply, the system reaches agreements that produce high quality solutions for the tickets received. This assumption has allowed us to perform more comprehensive tests with the small amount of data that we have and to check the advantages of following different dialogue strategies and of the amount of available knowledge about the preferences of other agents. However, a cooperative approach where agents are not self-interested and collaborate to reach the best agreement would be appropriate for this example and will be implemented and evaluated in the future.

In the current configuration of the example, we have considered that agents follow the same strategy during the entire dialogue. However, more efficient results could be achieved if they would be able to scan the environment and adapt their strategy as changes occur or to take

into account the actual behaviour of other agents. Future work will investigate more elaborated strategies to cope with these issues.

Given that agents participate in the argumentation dialogue in pairs, another issue to be discussed is the scalability of the approach concerning, for instance, the management of different interactions between distinct dialogues between pairs with a common agent. In our framework, agents make propositional commitments to the conclusions of the arguments that they propose during the dialogue. Therefore, an agent cannot present contradictory arguments to different agents (or to the same agent in different steps of the dialogue). This could happen if new information was updated in the agent's case-bases as a result of parallel conversations of this agent with different agents. Let us assume, for instance, that an agent has received an argument from another agent that includes a counter-example with a new domain-case that the agent does not have in its domain-cases case-base. Now, imagine that the agent learns this new domain-case in its domain-cases case-base. The new domain-case may allow the agent to retrieve it and generate a different solution (let us assume more suitable) for the problem at hand. Then, it could be the case that the agent was still engaged in a dialogue with an opponent agent attacking the position of the opponent, even when the opponent's position is the same as the new position that the agent can generate now with the new domain-case learned. Thus, the information conveyed in the argumentation dialogue would be inconsistent and the agent could violate its propositional commitments. Parallel dialogues with different agents can entail the introduction of incoherencies in the agents' case-bases and in the argumentation dialogue. In the current implementation of our argumentation framework, agents are not allowed to engage in parallel dialogues with different agents to avoid these undesirable situations. Furthermore, case-bases are not updated with new information until the whole argumentation process has finished and the final solution to be applied has been decided. Future work must investigate mechanisms to dynamically update the information of case-bases during the dialogue and extend the current approach to allow agents to participate in parallel argumentation dialogues.

Finally, due to the dynamism of the argumentation domain applied to open MAS, the contents of our case-bases can quickly become obsolete. Therefore, there is an important opportunity here to investigate new methods for the maintenance of the case-bases that improve the adaptability of the framework. In this research, we have followed the basic approach to update cases when a new case that is similar enough to an existent case in the case-base has to be added. However, we acknowledge that this can give rise to too large databases with obsolete cases that can hinder the performance of the whole system.

Funding

This work is supported by the Spanish government grants [CONSOLIDER-INGENIO 2010 CSD2007-00022, TIN2008-04446, and TIN2009-13839-C03-01] and by the GVA project [PROMETEO 2008/051].

References

[1] Aamodt, A., Plaza, E., 1994. Case-based reasoning: foundational issues, methodological variations and system approaches. AI Communications 7 (1), 39–59.

[2] Aleven, V., Ashley, K. D., 1997. Teaching case-based argumentation through a model and examples, empirical evaluation of an intelligent learning environment. In: Artificial Intelligence in Education, AIED-97. Vol. 39 of Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 87–94.

[3] Amgoud, L., Hameurlain, N., 2006. A formal model for designing dialogue strategies. In: 5th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS-06. ACM Press, pp. 414–416.

[4] Amgoud, L., Kaci, S., 2005. On the study of negotiation strategies. In: AAMAS 2005 Workshop on Agent Communication, AC-05. ACM Press, pp. 3–16.

[5] Amgoud, L., Maudet, N., 2002. Strategical considerations for argumentative agents (preliminary report). In: 9th International Workshop on Non-Monotonic Reasoning, NMR-02. LNAI. Springer, pp. 399–407.

[6] Amgoud, L., Parsons, S., 2001. Agent dialogues with conflicting preferences. In: 5th International Workshop on Agent Theories, Architectures and Languages, ATAL-01. LNAI. Springer, pp. 1–17.

[7] Artikis, A., Sergot, M., Pitt, J., 2009. Specifying norm-governed computational societies. ACM Transactions on Computational Logic 10 (1).

[8] Ashley, K. D., 1991. Reasoning with cases and hypotheticals in hypo. International Journal of Man-Machine Studies 34, 753–796.

[9] Atkinson, K., Bench-Capon, T., 2007. Practical reasoning as presumptive argumentation using action based alternating transition systems. Artificial Intelligence 171 (10-15), 855–874.

[10] Baruque, B., Corchado, E., Mata, A., Corchado, J. M., 2010. A forecasting solution to the oil spill problem based on a hybrid intelligent system. Information Sciences 180 (10), 2029–2043.

[11] Bench-Capon, T., Atkinson, K., 2009. Argumentation in Artificial Intelligence. Springer, Ch. Abstract argumentation and values, pp. 45–64.

[12] Bench-Capon, T., Sartor, G., 2003. A model of legal reasoning with cases incorporating theories and values. Artificial Intelligence 150 (1-2), 97–143.

[13] Bench-Capon, T. J., 1998. Specification and implementation of toulmin dialogue game. In: International Conferences on Legal Knowledge and Information Systems, JURIX-98. Frontiers of Artificial Intelligence and Applications. IOS Press, pp. 5–20.

[14] Black, E., Atkinson, K., 2011. Choosing persuasive arguments for action. In: 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-11). ACM Press, pp. 905–912.

[15] Devereux, J., Reed, C., 2009. Strategic argumentation in rigorous persuasion dialogue. In: 6th International Workshop on Argumentation in Multi-Agent Systems, ArgMAS-09. pp. 37–54.

[16] Dignum, F., Weigand, H., 1995. Communication and Deontic Logic. In: Wieringa, R., Feenstra, R. (Eds.), Information Systems - Correctness and Reusability. Selected papers from the IS-CORE Workshop. World Scientific Publishing Co., pp. 242–260.

[17] Dignum, V., 2003. Phd dissertation: A model for organizational interaction: based on agents, founded in logic. Ph.D. thesis, Proefschrift Universiteit Utrecht.

[18] Dung, P. M., 1995. On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming, and n -person games. Artificial Intelligence 77, 321–357.

[19] Heras, S., 2011. Case-Based Argumentation Framework for Agent Societies. Ph.D. thesis, Departamento de Sistemas Informáticos y Computación. Universitat Politècnica de València. http://hdl.handle.net/10251/12497.
URL `http://hdl.handle.net/10251/12497`

[20] Heras, S., Botti, V., Julián, V., 2009. Challenges for a CBR framework for argumentation in open MAS. Knowledge Engineering Review 24 (4), 327–352.

[21] Heras, S., Botti, V., Julián, V., 2010. On a Computational Argumentation Framework for Agent Societies. In: AAMAS 7th International Workshop on Argumentation in Multi-Agent Systems, ArgMAS-10. ACM Press, pp. 55–72.

[22] Heras, S., García-Pardo, J. A., Ramos-Garijo, R., Palomares, A., Botti, V., Rebollo, M., Julián, V., 2009. Multi-domain case-based module for customer support. Expert Systems with Applications 36 (3), 6866–6873.

[23] Jennings, N. R., Faratin, P., Lomuscio, A. R., Parsons, S., Sierra, C., Wooldridge, M., 2001. Automated negotiation: prospects, methods and challenges. International Journal of Group Decision and Negotiation 10 (2), 199–215.

[24] Kakas, A., Maudet, N., Moraitis, P., 2005. Modular Representation of Agent Interaction Rules through Argumentation. Autonomous Agents and Multi-Agent Systems 11, 189–206.

[25] Karacapilidis, N., Papadias, D., 2001. Computer supported argumentation and collaborative decision-making: the HERMES system. Information Systems 26 (4), 259–277.

[26] Liao, B., Huang, H., 2010. ANGLE: An autonomous, normative and guidable agent with changing knowledge. Information Sciences 180 (17).

[27] MacKenzie, J. D., 1979. Question-begging in non-cumulative systems. Philosophical Logic 8 (1), 117–133.

[28] Moore, D., 1993. Dialogue Game Theory for Intelligent Tutoring Systems. Ph.D. thesis, Leeds Metropolitan University.

[29] Ontañón, S., Plaza, E., 2007. Learning and joint deliberation through argumentation in multi-agent systems. In: 7th International Conference on Agents and Multi-Agent Systems, AAMAS-07. ACM Press.

[30] Ontañón, S., Plaza, E., 2009. Argumentation-Based Information Exchange in Prediction Markets. In: Argumentation in Multi-Agent Systems. Vol. 5384 of LNAI. Springer, pp. 181–196.

[31] Oren, N., Norman, T. J., 2009. Arguing Using Opponent Models. In: 6th International Workshop on Argumentation in Multi-Agent Systems, ArgMAS-09. Vol. 6057 of Lecture Notes in Computer Science. Springer, pp. 160–174.

[32] Perelman, C., Olbrechts-Tyteca, L., 1969. The New Rhetoric: A Treatise on Argumentation. University of Notre Dame Press.

[33] Pinzón, C. I., de Paz, J. F., Herrero, A., Corchado, E., Bajo, J., In Press. idMAS-SQL: Intrusion detection based on mas to detect and block sql injection through data mining. Information Sciences.

[34] Prakken, H., Sartor, G., 1996. A dialectical model of assesing conflicting arguments in legal reasoning. Artificial Intelligence and Law 4, 331–368.

[35] Rahwan, I., Larson, K., 2009. Argumentation and Game Theory. Argumentation in Artificial Intelligence, 321–339.

[36] Rahwan, I., Larson, K., Tohmé, F., 2009. A characterisation of strategy-proofness for grounded argumentation semantics. In: 21st International Jont Conference on Artifical Intelligence, IJCAI-09. Morgan Kaufmann Publishers Inc., pp. 251–256.

[37] Rahwan, I., Ramchurn, S. D., Jennings, N. R., McBurney, P., Parsons, S., Sonenberg, L., 2003. Argumentation-based negotiation. The Knowledge Engineering Review 18 (4), 343–375.

[38] Rahwan, I., Simari, G. (Eds.), 2009. Argumentation in Artificial Intelligence. Springer.

[39] Rahwan, I., Sonenberg, L., Jennings, N. R., McBurney, P., 2007. STRATUM: A methodology for designing heuristic agent negotiation strategies. Applied Artificial Intelligence 21 (6), 489–527.

[40] Rissland, E. L., Ashley, K. D., Branting, L. K., 2006. Case-based reasoning and law. The Knowledge Engineering Review 20 (3), 293–298.

[41] Rissland, E. L., Skalak, D. B., 1991. CABARET: Rule interpretation in a hybrid architecture. International Journal of Man-Machine Studies 34, 839–887.

[42] Rissland, E. L., Skalak, D. B., Friedman, M. T., 1993. BankXX: a program to generate argument through case-based search. In: 4th International Conference on Artificial Intelligence and Law, ICAIL-93. ACM Press, pp. 117–124.

[43] Roth, B., Rotolo, A., 2007. Strategic argumentation: a game theoretical investigation. In: Proceedings of the Eleventh International Conference on Artificial Intelligence and Law. ACM Press, pp. 81–90.

[44] Searle, J., 2001. Rationality in Action. MIT Press.

[45] Sierra, C., Botti, V., Ossowski, S., 2011. Agreement Computing. KI - Künstliche Intelligenz DOI: 10.1007/s13218-010-0070-y.

[46] Skalak, D., Rissland, E., 1992. Arguments and cases: An inevitable intertwining. Artificial Intelligence and Law 1 (1), 3–44.

[47] Soh, L.-K., Tsatsoulis, C., 2005. A real-time negotiation model and a multi-agent sensor network implementation. Autonomous Agents and Multi-Agent Systems 11 (3), 215–271.

[48] Sycara, K., 1990. Persuasive argumentation in negotiation. Theory and Decision 28, 203–242.

[49] Tolchinsky, P., Modgil, S., Cortés, U., Sànchez-Marrè, M., 2006. CBR and argument schemes for collaborative decision making. In: 1st International Conference on Computational Models of Argument, COMMA-06. Vol. 144. IOS Press, pp. 71–82.

[50] Yuan, T., Moore, D., Grierson, A., 2010. Assesing Debate Strategies via Computational Agents. Argument and Computation 1 (3), 215–248.

[51] Yuan, T., Svansson, V., Moore, D., Grierson, A., 2007. A Computer Game for Abstract Argumentation. In: Workshop on Computational Models of Natural Argument, CMNA-07. pp. 62–68.