Using Case Based Reasoning and Principled Negotiation to provide Decision Support for Dispute Resolution

Davide Carneiro [a], Paulo Novais [a], Francisco Andrade [b],
John Zeleznikow [c] and José Neves [a]

[a] *Department of Informatics, University of Minho, Braga, Portugal*
[b] *Law School, University of Minho, Braga, Portugal*
[c] *School of Management and Information Systems, Victoria University, Melbourne, Australia*

Abstract. The growing use of Information Technology in the commercial arena leads to an urgent need to find alternatives to traditional dispute resolution. New tools from fields such as Artificial Intelligence should be considered in the process of developing novel Online Dispute Resolution platforms, in order to make the ligation process simpler, faster and conform with the new virtual environments. In this work, we describe UMCourt, a project built around two sub-fields of Artificial Intelligence research: Multi-agent Systems and Case-based Reasoning, aimed at fostering the development of tools for Online Dispute Resolution. This is then used to accomplish several objectives, from suggesting solutions to new disputes based on the observation of past similar disputes, to the improvement of the negotiation and mediation processes that may follow. The main objective of this work is to develop autonomous tools that can increase the effectiveness of the dispute resolution processes, namely by increasing the amount of meaningful information that is available for the parties.

1. Introduction

In the words of Abraham Lincoln: "Discourage litigation, persuade your neighbour to compromise where you can. Point out to them how the nominal winner is often the loser... in expenses and waste of time". This is one of the sentences that better describes the need for an alternative to traditional courts and the appearing of Alternative Dispute Resolution (ADR) [1]. ADR includes mechanisms that aim to solve disputes without recurring to the traditional judicial process, i.e. courts. ADR methods are nowadays broadly used by both the legal system and the parties involved as the first step in the path towards the dispute resolution. There are even countries in which parties are encouraged or required to try an alternative way of dispute resolution before advancing into a court.

One of the factors that definitively boosted the acceptance of ADR is the current workload of courts. Often it takes many years for a case to receive a final hearing. Thus, a new approach is required to diminish the number of cases that actually need to be resolved by traditional court processes. Parties, in particular, also look at ADR tools in search for lower expenses, more privacy and less exposure than the provided by traditional trials. Another advantage is that the entity that will decide a given case can be agreed on by the parties instead of being mandatorily assigned, which in a certain way increases the confidence and overall satisfaction of the disputing parties in the result obtained.

An alternative way of solving disputes arising out of the contractual performance in virtual environments is that of Online Dispute Resolution. This new paradigm allows the moving of the traditional alternative dispute resolution methods from a physical to virtual place [24]. Hence,

parties not only reduce litigation but gain a simple and efficient way of dealing with disputes, saving both temporal and monetary costs [33]. Several methods of ODR may be considered, from negotiation and mediation to variations of arbitration or modified jury proceedings [34].

Indeed, the growing use of Information and Communication Technologies (ICT) in the commercial arena lead to an urgent need to find alternatives to dispute resolution. The traditional paper-based courts and arbitration, designed for the industrial era, are now outdated. In fact, Civil and Commercial Law are still turned towards a type of society and a way of doing business that is related to a mostly industrial society and the traditional use of the paper as the primary support tool. Given the rising use of Information Technology, electronic means for resolving disputes need to be considered. This should include emerging tools from fields such as Artificial Intelligence that are used for effective idea and strategy generation. This will result in a more creative process, rich in ideas and solutions, that will ultimately lead to simpler, faster and more effective dispute resolution processes.

In this sense, the use of software agents that embody intelligent techniques can become particularly interesting considering their ability to recognise and evaluate facts, positions and relevant information. Moreover, the use of these agents, as building blocks of distributed platforms that support ODR processes, has to be considered. Of major importance in this context are systems that are able to simulate and calculate outcomes of disputes, establish negotiation paths to achieve the desired objectives and warn the parties about the possible alternatives to an agreement.

There is a variety of ODR systems, including legal knowledge based systems and systems that help settle disputes in an online environment [35]. Second generation ODR, in which ODR systems act as an autonomous agent [36] are a most appealing way of solving disputes. Also interesting here is the view of Katsh and Rifkin concerning the role of technological tools. The authors see technology that works with the mediator or arbitrator as the fourth party in the dispute, together with the two disputing parties and the third neutral [37]. In fact, there has been a recent tendency to foster the intervention of software agents, acting either as decision support systems [24] or as real electronic mediators [36]. Surely, this latest role for software agents could benefit from the use of Artificial Intelligence techniques such as Case-based Reasoning (CBR) and information and knowledge representation models. Models of the description of the fact situations, of the factors relevant for their legal effects allow the agents to be supplied with both the static knowledge of the facts and the dynamic sequence of events [36].

Evidently, representing facts and events is not sufficient for a dispute resolution. In order for the software agent to perform actions of utility for the resolution of the dispute, it also needs to know the terms of the dispute and the rights or wrongs of the parties [36]. Moreover, it is of utter importance that the software agent is able to foresee the legal consequences of the said facts and events. We thus need to consider whether agents can evaluate the position of the parties and present them with useful proposals, taking into a consideration of which of the two parties would have a higher probability of being penalised or supported by a judicial decision of the dispute and, therefore, who would be more or less willing to make concessions in their claims [16]. The ability to understand the position of the parties is vital for the successful involvement of software agents in the process. To do so, it is mandatory for the software agent to have the characteristics of consistency, transparency, efficiency and enhanced support for dispute resolution, in order to allow it to replicate the manner in which decisions are made and thus make the parties aware of the likely outcome of litigation [3]. That is to say, software agent intervention in ODR procedures should take into account the alternatives, for the parties, to an ODR negotiated agreement.

Thus, in such a context, it would be interesting to consider some useful and well known ADR concepts such as the BATNA, or Best Alternative to a Negotiated Agreement. This concept denotes the best scenario possible if the negotiation process fails, i.e., if the case was to go into a court, what would be the best outcome? Knowing their BATNA helps parties to achieve an outcome using negotiation that is hopefully better than it would be if achieved through litigation [45]. The importance of this concept is well expressed by Goldberg et al. in [31]. In fact, if a party is unaware of what results could be obtained if the negotiations are unsuccessful, he runs

the risk of: entering into an agreement that he would be better off rejecting; or rejecting an agreement he would be better off entering into.

Following the same line of thought the WATNA, which denotes the worst possible outcome along a litigation path, should also be considered [32]. It can also be quite relevant in complementing principled negotiation [46] with a justice or rights based approach, and thus leading to a calculation of the real risks that parties will face in judicially determined litigation, imagining the worst possible outcome. In the same sense, considering the MLATNA – Most Likely Alternative to a Negotiated Agreement, may also be useful as a way to show parties the alternative that is most likely to happen if the negotiated process fails. These three concepts are definitively of utter importance for the parties to have a clear general picture of what the alternatives are, thus supporting wiser decisions. Finally, one last important concept is the one of ZOPA – Zone of Potential Agreement, which denotes the range of the possible outcomes, i.e., what may happen at the end of the process, and is delimited by the WATNA and BATNA [4].

Considering the usefulness of these concepts in ODR, it becomes obvious that these systems will go much further than just transposing ADR ideas into ODR environments. They should actually be guided by judicial reasoning, getting disputants to arrive at outcomes in line with those a judge would reach [14]. Despite there being difficulties to overcome at this level, the generalised use of software agents as Decision Support Systems points out the usefulness of following this path.

Taking all of this into consideration, the work described in this paper focuses on two key points. First, the use of Artificial Intelligence techniques that enable not only contextualized information retrieval but also allow the system to evolve according to the results of this retrieval. Particularly, we will be analyzing the role that Case-based Reasoning can play not only in the information retrieval task but also on improving other processes. Second, we will be looking at the use of traditional techniques, specifically negotiation and mediation, as a way to improve the efficiency of dispute resolution. These two trends merge to fulfill a unique objective: to provide contextualized information and guidance processes to disputant parties so that they can solve their dispute out of courts.

1.1 Related Work

Proving the validity and relevancy of this research field, a group of interesting projects with the objective of giving birth to dispute resolution mechanisms can be pointed out. In this section we describe some of the projects which intersect with this work.

Rule-based Legal Decision-making System (LDS) dates from 1980 and was one of the first negotiation support systems to be developed [50]. The domain of this system was liability law. This field of law holds responsible product distributors and manufacturers for the injuries their products may cause. The system created embodied the skills and knowledge of a human expert in the shape of antecedent-consequent rules. The project had the objective of formalizing the decision-making processes of attorneys and claims adjusters involved in product liability litigation in the shape of rule-based models so that the effects that changes in legal doctrine have in settlement strategies and practices could be studied. The authors formalized the strict-liability concept on ROSIE language, so that the defendant could or could not be considered liable.

A more recent project, focusing on providing support for decisions is EXPERTIUS. This is a decision-support system that advices Mexican novice judges and clerks upon the determination of whether the plaintiff is or not eligible for granting him/her a financial pension (on the basis of the "feeding obligation") and if so upon the determination of the amount of that pension [51]. The system has three main modules: a tutorial module, an inferential module and a financial module. The tutorial module guides the user through the accomplishment of several tasks. The inferential module evaluates evidence based on weights that the user assigns to each piece of evidence. It determines which prepositions are defeated and which prevail. At last, the financial module assists the user on the calculus of the value of pensions according to some criteria.

For reasoning in these terms, EXPERTIUS has an extensive way of representing the knowledge about the several parameters. Judicial expert knowledge was represented as having

three interrelated levels: one for representing the expert knowledge, one for representing the decisions internal to each procedural stage as regulated by procedural law and a third one that corresponds to the dialogical confrontation pattern of the case that arises simultaneously to the decisions taken in the intermediate level.

The knowledge embodied in the system was divided into five layers. The *formal doctrine* contained rules from the legislation and common law. The *informal principles* contained rules that are not explicitly expressed in the law but are generally agreed upon by legal practitioners. Under the *strategies* layer the authors coded the methods used by legal practitioners to accomplish a given goal. The *subjective considerations* layer contains rules that anticipate the subjective responses of people involved in legal interactions. At last, the *secondary effects* layer contained rules that describe the interactions between rules. The authors concluded that despite the number of rules needed for formalizing the law and the strategies, the rule-based model was feasible and suited for this particular domain.

In the field of negotiation support, a major work is the Family_Winner project, which integrates game theory and heuristics in the field of family law, more specifically, in disputes arising from divorce processes [2]. The research is largely based on the Adjusted Winner algorithm [2], an algorithm that merges insights from the fields of game and decision theory. This algorithm aims at the distribution of a group of items or issues by the parties, according to a mechanism of point allocation. In this process, each party quantifies how much he/she wants the item and whoever values more a given item receives it.

Still in the negotiation support, another project worth mentioning is SmartSettle [48]. It is an online negotiation system that can be described as a generic tool for decision-makers. It is intended for parties with conflicting objectives that wish to reach a formal agreement. For achieving it, tools that help to define the interests and possible trade-offs are used, with an emphasis on recognizing the parties' satisfaction in the search for optimal global solutions. This platform can be used to solve problems relating to family, insurance, real estate, labour-management, contract negotiations, among others.

When comparing the work presented in this paper with the related work mentioned in this section, the main difference that can be noted is that, contrary to the most common approach, we do not aim to provide support for dispute resolution in court. We rather aim for providing valuable information for the parties and eventually the neutrals, so that better and more informed decisions can be taken. While doing so, we also expect disputing parties from going into court, by making them aware of the possible consequences, fostering cooperative off-court processes. Nevertheless, the tool presented can still be used throughout a complete dispute resolution process, ranging from the initial providing and compilation of information, to the definition of an outcome. Another major difference is that in this work, there is a close integration between the phase of generating information and the phase of actual dispute resolution through traditional means. That is, the information that is compiled to inform the disputing parties is also used to support and help define the mediation or negotiation process that may follow.

2. Agents and Negotiation

As mentioned before, there is a particular interest in implementing ODR tools that behave the same way a human would. Particularly interesting is the use of software agents to implement social interaction processes, specifically negotiation. Generally, negotiation is classified as distributive or integrative [5].

In distributive negotiation, one looks at the problem as something that can be divided and distributed by the parties in an attempt to maximize their satisfaction. An example scenario is the classical winding up of a company in which the assets of the company are sold and the proceedings collected are used to discharge the liabilities. In game theory this is known as a zero-sum game. Two important concepts here are the ones of utility and resistance [6]. Utility denotes the value that a given item has to a party while resistance denotes the willingness of a party to change the utility of an item. A good negotiator usually tries to convince the other party

that certain items do not have the value that they are given. The negotiator will succeed if the opponent has a low resistance in that item and if he does so, it will be easier for the negotiator to win that item or he will, at least, be in a better position for the rest of the negotiation process [7]. Accordingly, one can define utility functions that help to understand how each party values the items being distributed, therefore predicting possible outcomes and the evolution of the negotiation process [8].

In integrative negotiation, the problem is expected to have more solutions than the ones visible at first sight. In these types of problems, the parties try to bring to the table as much interests as possible so that there are more and more valuable items with which to negotiate. When the parties are increasing the value of what they put in the table, they take into account their interests which include the needs, fears, concerns, and desires. This type of negotiation is also known as interest-based, as parties try to combine their interests and find common points in which they are satisfied. By doing so, more satisfactory outcomes are achieved by both parties. This makes integrative negotiation more desirable than distributive.

A good example for illustrating this difference is an old story about a brother and a sister both wanting the last orange. To solve the dispute, their mother initially gives both children half the orange. When she asks them their goals, she finds that the brother wants the orange for the juice to drink while the sister wants the rind to flavor an orange cake. So the mother gives the sister the rind and the brother the rest and they both get what they desire [49]. The outcome is optimum. An important concept in this field is the one of Pareto efficiency [9]. In this case, the solution obtained with the integrative approach would have been a Pareto efficient solution as no better solution could have been achieved.

A key factor in dispute resolution is thus, as seen in the previous example, to identify the interests of each party so that their positions can be better understood. Such a process can eventually lead to what is known in game theory as a win-win game, i.e., all the parties are better at the end of the negotiation process than when it started.

2.1. Principled Negotiation

One of the most effective methods for resolving conflicts is through negotiation. Specifically, Principled Negotiation was developed by the Harvard Negotiation Project [46] and focuses on the notion that parties look for mutual gains. When interests conflict, Principled Negotiation advocates parties arrive at a ruling that is independent of the beliefs of either side. Principled Negotiation puts forward five key points:

1) *Separate the people from the problem*. Personal matters or stronger personalities can often influence the outcome of dispute resolution processes. This is especially significant in scenarios in which the disputant parties have or had a personal relationship, such as in family law disputes. Solving disputes by leaving aside personal affairs can avoid biasing a solution towards a more influencing party.
2) *Focus on interests, not on positions*. It is important for a party to understand what their interests are rather than blindly defending a position that may not be that solid. By isolating the reasons why a position is most appealing, participants in a negotiation will increase the chance of achieving agreement.
3) *Invent options for mutual gain*. Although parties have divergent objectives or interests, they might still find positions that encompass mutual gain. Once interests have been ranked to determine the relative importance of each, a range of options is discussed before deciding on an outcome. Next, the negotiators need to invent options for mutual gain.
4) *Insist on objective criteria*. There are negotiation scenarios that cannot be treated as a win-win situation. In such cases, unbiased independent evaluations should provide an outcome or solution that both parties will agree on.
5) *Know your Best Alternative to a Negotiated Agreement*. When two parties negotiate they aim at achieving a better result than would otherwise occur. Being unaware of what results one could obtain if the negotiations are unsuccessful, one runs the risk of:
   a. Entering into an agreement that would be better off rejecting; or

b. Rejecting an agreement that would be better off entering into.

All the work presented in this paper is guided by these five principles. They were thus present throughout the development process of UMCourt, an Online Dispute Resolution platform, which will be described in the following section.

3. UMCourt

UMCourt is being developed at the University of Minho, in the context of the TIARAC project (*Telematics and Artificial Intelligence in Alternative Conflict Resolution*). This project aims to explore the potential of the combined use of new technologies with Artificial Intelligence techniques to Law in the domain of Alternative Dispute Resolution [47]. UMCourt consists of a multi-agent system, based on open standards and technologies. It aims at more than a simple legal information management tool, being its core a group of agents that embody intelligent techniques for problem solving. The resulting platform is able to provide services such as determining possible outcomes and respective likelihood, proactively providing relevant information in real time, automatic creation of legal documents, determination of the major concepts in ADR for each dispute, among others. Law experts work in close cooperation with the development team of UMCourt so that the resulting platform addresses the above challenges.

These challenges are often faced in the legal domain as well, namely the definition of standards for legal information representation or the formalization of argumentation. Ashley, in [13], even argues that "Perhaps, the most interesting opportunity for the judicial and AI & Law communities *(…)* is in cooperating in the design of standards for the presentation of factual descriptions and discussions of law in case opinions.". There is also the need to provide this information to the parties in an intuitive fashion as there is, not infrequently, a pronounced difficulty in understanding legal information. This evidently constitutes a disadvantage that may be aggravated if the party does not know how to efficiently work with the ODR tool. There is thus an obvious interest in this research field from both legal and informatics practitioners.

3.1. Portuguese Labour Law and Alternative Dispute Resolution

The work described here was developed in the context of the Portuguese labour law, focusing especially on the relation between employers and employees. Particular attention is paid to the scenario of an employee being fired, a case in which litigation will certainly occur.

In firing an employee, a variety of considerations need to be examined. First of all, litigation is a quite slow process, especially in Portugal, and these results in several disadvantages for both parties [10]. In a dynamic economy it is important, both for employers and employees, to have supporting mechanisms to settle the disputes in a fast and efficient way. ADR has long been considered and practiced, mainly negotiation by Lawyers. Indeed, the Portuguese Labour Code expressly recognises the possibility of the parties submitting their disputes to arbitration (art. 564º Labour Code) [11]. But it must be recognized that the normal way of solving this kind of disputes is through conciliation by the Judge, just before trial, and after a long procedure.

Under legal systems such as the Portuguese, in the case of an employee being fired, a huge deal of legal parameters have to be considered: the possibility of a "just cause for dismissal" being declared by the Court, the existence (or not) of a valid and legal procedure of dismissal, the validity (or not) of the dismissal, the antiquity of the worker in the company, supplementary work, night work, justified or unjustified absence from work, the possibility of dismissal being accepted without indemnities or of it being accepted but accompanied by indemnities that could range from a very low to a very high amount of money [10, 11]. The company performing the sacking can either pay a small amount of money, thus making it worthwhile to fire the employee, or it may be placed in the position of being forced to pay a considerable sum. For the worker, the amounts involved are significant. Being fired without good indemnities may lead to the loss of a job and no payment as compensation. But it could, on the other side, become

beneficial for him to be fired, provided that he receives a significant cash offering. For the parties in a labour conflict, the calculation of the possibilities of an outcome of litigation is vital. Thus it is vital for both parties to know in advance their best and worst alternatives to a negotiated agreement.

3.2. The Architecture of UMCourt

The conflict resolution platform is being developed according to the traditional client-server model. The human user is able to interact with the platform by means of a standard web browser. This constitutes the client. Additionally, mobile versions of the interface have been developed, which allow users to access a more reduced set of functionalities, while on the move. The server relies on the Multi-agent System paradigm [14]. Under this paradigm, a group of intelligent agents share information and (possibly) cooperate to accomplish a common objective. Wooldridge and Jennings define a weak notion of agent in [15] as an agent that shows autonomy, social ability, reactivity and pro-activeness. A stronger notion of agent may involve features like mobility, veracity, benevolence or rationality. Such characteristics can thus significantly enrich platforms developed under this paradigm.

However, more than just choosing a technology or a paradigm, one needs a methodology. Wooldridge et al. propose a methodology in [16] for the development of agent-based applications that begins with a high level definition of the system in terms of their roles. Following this methodology, we have defined four main groups of agents according to their roles: *main*, *secondary*, *interface* and *control*. Agents in group *main* have important decisory roles, typically empowered with autonomy and enhanced communication skills. Agents in group *secondary* have as main role to support the lifecycle of the remaining agents by providing low-level but vital services. Agents that belong to the *interface* group are used to act as interface between a user in the web page and the multi-agent system. Finally, agents in group *control* are responsible for the correct lifecycle of the whole platform. Given the scope of the paper, a thorough description of the agents is not provided. We will rather focus on the agents directly involved in the work described in the following sections. This is detailed in Table 1. For a detailed description of the architecture please see [52].

Table 1. The main agents that build up the architecture, given the focus of this paper, with a description of their roles.

Agent name	Role
Coordinator	This agent coordinates the remaining agents in order to implement higher level tasks through the concatenation of simpler ones. In order to do so, the agent contains knowledge about the high level tasks defined in terms of finite state automata. A typical scenario is the coordinator receiving external requests for given tasks, which it will answer by combining tasks from other agents.
Database	The Database agent implements a transparent bridge to the database. Thus, when the remaining agents need to interact with the database they do it by means of requests to this agent and not directly. This has as main advantage to isolate the database from the platform.
Evaluator	This agent collects information about key operations of the platform so that later its performance can be assessed. Based on the information collected and on a description of the operations, the agent is also able to issue recommendations for the improvement of the performance.
NearestNeighbour	This agent implements a nearest neighbour algorithm with the objective of defining a measure of similarity between two cases.
TemplateRetrieval	This agent implements several information retrieval algorithms, from which the remaining agents can choose, depending on their objectives.
Utility	The Utility agent is able to determine the utility of a given case for a given party, i.e., to measure how good a given case is to a party. Typically, this is used to determine the utility of past known cases to a party which is now dealing with a similar one.
Blackboard	This agent implements a mediation and a negotiation algorithm as well as a shared space for message exchange. It is responsible for controlling the evolution of the processes.
Party	The party agent is the virtual representation of a disputing party in the platform. Therefore, different instances of this agent may be present at one time, each one representing a different party. This agent also establishes the bridge between user and platform, i.e., the user requests actions or sends messages through this agent.

Regarding the agent platform, an analysis of several frameworks has been conducted. Given their characteristics as well as our past experience, the choice has been in favour of the Jade (Java Agent Development Framework) platform [38]. Jade is a software framework that significantly facilitates the development of agent-based applications in compliance with the FIPA specifications [39].

Before advancing into further details, let us now make a first high level description of the main functionalities implemented by the described software agents. In these functionalities there is a close integration between the information retrieval and its use on the posterior processes (e.g. utility evaluation, negotiation, mediation), independently of the purpose. Some of this functionalities are simple ones (i.e. it takes only one agent and one iteration to implement it) while others are complex (i.e. it may take the interaction of more than one agent or more than one iteration of the same agent to implement it). A detailed description of these functionalities is given further ahead.

- *Parsing and Indexing* - In an ODR platform, information from a dispute that took place in the real world must be acquired. This generally includes, in a first instance, the parsing of documents to determine their syntactic validity. Afterwards, cases may be indexed so that they can be more efficiently retrieved. This platform is able to automatically parse cases by means of a SAX parser (Simple API for XML), determining their validity against the defined Schemas. Moreover, cases are also indexed in the database, enabling several tasks to be performed quicker.
- *Information Retrieval* - Information retrieval is a major concern in any knowledge-based domain. The legal one is evidently not an exception. It is especially important as it is the basis for a wide range of operations and decisions. In that sense, the UMCourt platform considers several information retrieval methods that allow for important information to be retrieved in a context-aware fashion.
- *Case Similarity Assessment* – From the point of view of a disputing party, the assessment of the similarity of his case with a past known case is very important. This will allow him to determine, for example, the likeliness of an outcome, based on an observation of the outcomes of past similar cases. Likewise, legal practitioners can also use such functionalities for training or for supporting their decisions. Thus, in UMCourt we have implemented several methods for similarity assessment.
- *Utility Computation* – Of major importance is also the ability to compute the utility of a given case. The issue here is to determine how good a given outcome is for a party, that is, if a given outcome was to occur, how much would a party win or lose. This is essential for a disputing party to take good and rational decisions.
- *Generating Important Information* – As mentioned above, there is a wide range of information that can be important for the parties at the time of taking decisions. The ability to compile all this information is thus of value and was considered in the development of UMCourt.

The use of these more basic functionalities enables or fosters the implementation of higher level ones, which significantly improve the experience of end-users, whether they are the disputing parties or the legal practitioners.

- *Case-based Reasoning* – The use of CBR techniques can be of importance for both the provision of information to the users and for the evolution of the platform itself. In fact, platforms based on the CBR paradigm can evolve in a dynamic and autonomous way, learning with the own correct and incorrect decisions. Moreover, these platforms can select past cases that may relevant to solve a given problem and are thus of value for the disputing parties as well.
- *Negotiation and Mediation* – Simply suggesting a solution or an outcome for a solution is generally not enough, i.e., parties will most likely not agree on it. There is thus the need to let the parties change that solution or negotiate a new one in which they agree. It is thus

important to consider successful dispute resolution methods such as negotiation or mediation. In UMCourt this is done with a particular emphasis on the transparent integration with the information that is retrieved, i.e., the same information that is used to inform the disputing parties is also used to guide negotiation and mediation processes.

4. Case-based Reasoning in UMCourt

As stated above, the estimation of possible outcomes is an important problem as it is the base for a supported decision and planning process in dispute resolution. In this sense, there are several approaches that can be considered. Utility functions have considerable applications in the legal domain (see for example [17], [18] and [19]). A good example is the above mentioned Family_Winner system, in which Bellucci and Zeleznikow have applied this technique to family mediation [20], after they observed that an important way in which family mediators encourage disputants to resolve their conflicts is through the use of compromise and trade-offs [21].

Artificial Intelligence techniques have also been used to address the same challenge. James Popple used a simple Expert System to provide advice in fields of case law that have been specified by a legal expert in a specification language defined by the author [25]. Split-Up combines rule-based reasoning with neural networks in order to build a system to assist parties involved in property settlements in an Australia divorce process [26]. UMCourt is framed in this group of systems that apply AI techniques to ODR.

In the Case Based Reasoning paradigm, cases represent past experiences stored in memory. Each case contains the description of the problem that was encountered (the initial state of the world), the solution adopted to deal with it, and the outcome (the final state of the world). As this case memory grows, the system gains more knowledge about the problem and the possible different ways to deal with it. In each new problem that is faced, past cases are analysed in search for similarities, and solutions of similar cases are adapted to be used to solve the new problem. At the end of this process, it is possible to learn with the success or failure of the application of the solution to the new problem.

Choosing a case-based approach was not an arbitrary decision. In fact, law itself relies on the concept of case and implements a very similar concept: the legal precedent [27]. This concept is defined by the Oxford dictionary as “a previous case or legal decision that may or must be followed in subsequent similar cases”. There are legal domains in which precedents are divided into two types: binding and persuasive precedents. The first type is generally the result of a process in a higher court and means that all lower courts must honour it. The persuasive precedent, on the other hand, arises from cases that are decided in lower courts and does not represent an obligation. However, in Portugal, as in civil law countries in general, this separation does not exist as all the precedents are simply persuasive. Judges may follow a precedent decision, but they are also entitled to decide otherwise, since judges are only bound to apply the legal norms and interpretation differences between judges may occur [45, 28].

A CBR model would, at first sight, be more suited to be used in common law: a law that is developed through decisions of courts. This, however, does not mean that it cannot be used in a civil law context, such as the Portuguese one, in which laws are written by a legislature's enactment [28]. It is our conviction that CBR is an appropriate method for such a problem-solving domain as the cases and their outcomes are often very similar to known past occurrences. This conviction is shared and supported by several authors. Ashley, in particular, poses the question: “should researchers in a civil law jurisdiction pursue work on implementing AI and Law models of case-based legal reasoning in a civil law context?”. He then answers the question with a conclusive “the answer may well be, “Yes”!” [13].

In that sense, throughout this section we will be guided by five main questions synthesized in [41] and, by answering to each one of them, define our agent-based CBR system, namely:

1) What makes up a case? How can we represent case memory?
2) How is memory organized? What are the indexing rules?

3) How does memory change? How do the case memory and indexing rules change over time?
4) How can we adapt old solutions to new problems? What are the similarity metrics and modification rules?
5) How can we learn from mistakes? What are the repair rules?

We commence by developing some basic concepts that will later be useful for understanding the CBR process model and the role of each of the presented agents.

4.1. Preliminary Considerations

Let us start with a high level description of the CBR process as used in our specific domain of application. There is a database of cases, which embodies information from legal disputes settled by courts, with their respective outcomes. The CBR process is mainly intended to look at these cases and infer what would happen if the two parties currently involved in the dispute, would have the matter resolved by a court. This basically consists in selecting the most similar cases and presenting their outcomes to be analyzed by the disputing parties. More than that, the system compiles information that is information for the decision making processes, including the previously mentioned BATNA, WATNA, ZOPA and MLATNA. With this information, parties can build their own image of the whole problem and of the possible paths and respective consequences.

At the end of this process, parties may choose to accept the proposed outcome or they may reject it. If they accept it, the value of the proposed case is increased, denoting that it was successfully applied. Alternatively, if the parties reject the proposed outcome, that value is decreased and, should the parties advance to court, a new case is learned by the system containing all the information that had already been provided by the parties and the outcome from the court. Otherwise, parties can also make use of a traditional dispute resolution method (e.g. negotiation, mediation), integrated in the UMCourt platform.

Parsing and Indexing

In an indexing process, property values are assigned to cases so that they can be easily identified according to those properties. Therefore, an important part of this process is the one of selecting the right properties for indexing the cases. Evidently, this is, once more, domain-dependent. In our case, we index the cases according to:

- Background information;
- Norms addressed by parties (i.e. legal information);
- Objectives of the parties;
- Date of the case;
- Outcome.

This allows agents to make requests to the database such as “all the cases in which norm X is addressed”, “all the cases that are within a given time frame” or “all the cases with a given outcome”. Our objective in indexing the cases has essentially to do with efficiency issues. It would be a rather time-consuming task to constantly search all the files for some parameters. By indexing them, this task is simplified to querying the database for the identifiers of the cases that match the desired criteria.

Indexing is often looked as a challenge to overcome in CBR systems, mainly when the indexing method is manual as it tends to become a slow and hard task. Automated methods are thus preferred. In the case of this work, indexing is an automated process. An indexing agent frequently monitors a predetermined folder in the system and indexes new files as they are added. However, the validity of their structure must first be determined.

This agent therefore starts by using a SAX parser to parse the content of the new file and checks if it is valid according to the current Schemas. If it is, the case is indexed. The indexing process is as simple as reading the appropriate contents from the parser result and add the

identifier of the case to the appropriate tables in the database. However, only the information that important for the retrieval of the cases is stored in the database while the remaining is kept in the files. All the information needed for indexing the case has been previously provided by the parties, while filling in the information requested, or is automatically inferred by the system (e.g. dates, norms addressed), ensuring that the indexing process itself is a completely automated one.

Additionally, a record of failed cases is also maintained, in which each case is associated with the reasons for failure. These cases are intended to be looked at when the CBR process fails in the search for reasons for the failure, as will be seen below. Moreover, when the Retriever agent is retrieving the cases from memory it may also look at this record before taking actions that influence the parameters of the search (e.g. changing the similarity threshold), in order to prevent errors that have already occurred in the past.

Once the information is indexed, a wide range of operation becomes possible. As an example, it is possible to mine the database for association rules [57], which are then used to classify the cases (Figure 1) [54]. This is useful for retrieving cases, as will be seen below, by selecting all the cases that belong to the same category or group, i.e., all the cases for which a given rule mined is true.

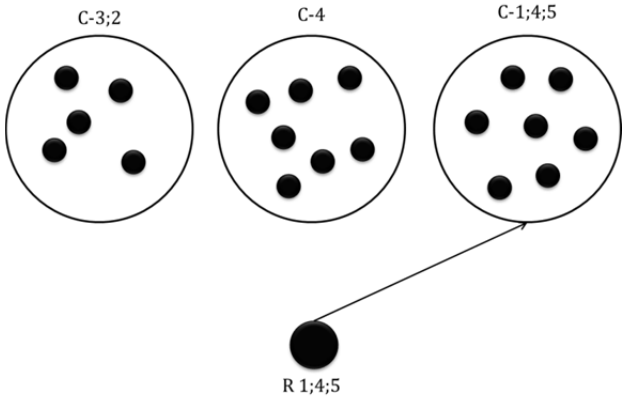


Figure 1. Indexed cases can be organized by means of association rules: all the cases for which a given rule is true belong to the same category.

It is also possible to quickly build different interpretations of the information, suited for specific applications. One of these interpretations is the representation of the information as vectors of binary entries. This is a fairly simple algebraic model for representing text documents in which instead of using textual fields, a case is represented as a vector. Specifically, in the context of this work, a case is seen as a vector V of binary entries, in which each entry $i < N$ corresponds to a fixed descriptor from the descriptor vector D of size N. Thus, the value of each binary entry denotes the presence or absence of that descriptor on the case. Descriptors denote important components of a case (e.g. legal norms, objectives of the party, winner of the case). Thus, one can look at a vector which represents a case and, considering the descriptors vector D, determine which information is or is not present on the case (Figure 2).

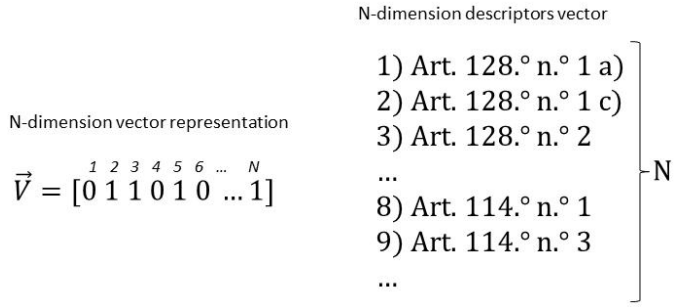


Figure 2. N-dimension vectorial representation of a case.

Basically, this representation of a case allows seeing which norms are addressed by each party, which are their objectives and what is the outcome. It is thus a very concise way of representing all this information, demanding very few resources to handle and to store. Following the same line of thought, a database with *m* cases in which each case is described by *N* descriptors can be represented as an *m*-by-*N* matrix in which each line is a vector representing a case. (Figure 3).

$$DB = \begin{matrix} \begin{matrix} 1 & 2 & 3 & \dots & N \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 1 & \cdots & 1 \end{bmatrix} \end{matrix} \begin{matrix} 1 \\ 2 \\ 3 \\ \dots \\ N \end{matrix}$$

Figure 3. Representation of a database of cases as a matrix.

Case Structure

In the CBR paradigm, a case is the basic unit of information. It represents a past experience and is thus a contextualized piece of knowledge. CBR allows us to estimate the outcome of a current experience by looking at a past similar experience (that took part in an equivalent context) and its respective outcome. The definition of the information that should be contained in the case is therefore a major task in the development of a CBR model. Generally, the information contained in a case can be organized into three distinct categories [42]:

- The Problem – The description of the problem, including the context of applicability, the initial state of the world and all other information that may help to define the problem.
- The Solution – A description of the list of steps that were taken in order to solve the problem. The description should be exhaustive enough to be applied again in an autonomous fashion and abstract enough to be adapted to new problems.
- The Outcome – The consequences of the application of the described solution, i.e., the final state of the world.

Whereas these categories do depend on the problem domain, there may be applications which consider only the problem and the solution (in which it is possible to derive solutions to new problems) while others consider the problem and the outcome (making it possible to estimate outcomes to new problems). In our particular case we consider the three categories, namely:

Table 2. The main agents that build up the architecture, given the focus of this paper, with a description of their roles.

Category	Information Type	Description
Problem	Background	Basic information about the parties and the dispute such as party's personal information and location, dispute starting date, witnesses.
	Objectives	A list of the initial objectives of each party towards the dispute, i.e., the expected outcome.
	Legal	Legal information such as the laws and norms addressed by the parties and witnesses to support their claims or the guilty statement.
	Dates	All the important dates of the case.
Solution	List of actions	A list of the actions performed by the parties in order to achieve the outcome. Generally, these actions comprise trade-offs.
Outcome	Outcome description	A list of items that describe the outcome in terms of indemnities to be paid, among others.
	Value	A value denoting the percentage of successful applications of this case in the dispute resolution process

Being more specific, cases are precisely defined using the Extensible Markup Language standard (XML). In that sense, we formally define the structure of the case using the XML Schema language and, consequently, the cases are XML files stored in the file system. Using XML allows us to take advantage of other related specifications, namely XML Encryption, XML Signature or XSL Transformation. Given the size of the XML Schemas that define the cases, these are not detailed here.

4.2. The CBR Process Model

We have, until now, described how the experiences are acquired, organized and stored in the system. However, purely storing information is not enough. We need to define the processes that acquire and use this information. To define this process, we have looked at the work of [30] and [40], and made made the necessary revisions in order for it to be used in our domain. It is a four step model composed by the Retrieve, Reuse, Revise and Retain phases (Figure 4).

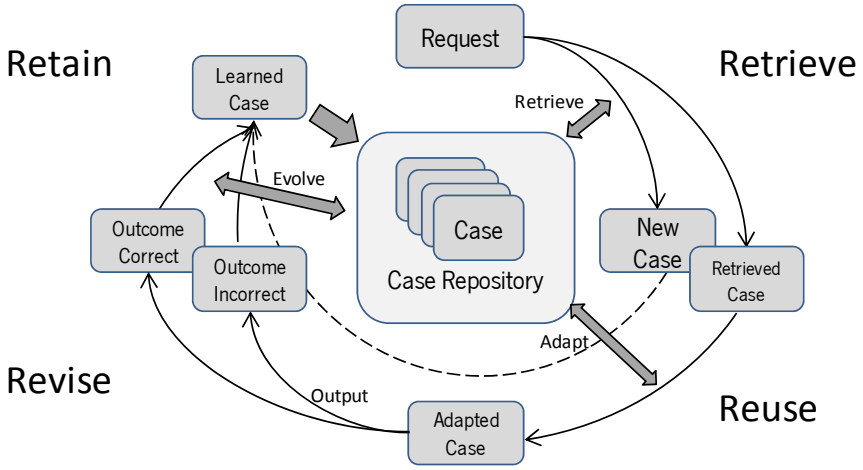


Figure 4. The CBR model implemented.

Retrieve

The process of retrieving may be triggered under different scenarios. It consists of selecting the cases from the case memory that are of relevance for solving a problem, by looking at their indexes. Unlike database searches that target a specific value in a record, retrieval of cases from the case base must be equipped with heuristics that perform partial matches, since in general there is no existing case that exactly matches the new case [43]. There are several techniques for retrieving cases. Our approach is a hybrid one and consists of two different phases [44]. In the first phase, a pre-selection of cases is performed. On the second phase, the pre-selection is evaluated, eventually more refined using similarity functions, and the cases are sorted. In each of these phases, two different algorithms can be combined. Each of these algorithms is analyzed in terms of their performance in the Results section.

In the pre-selection phase, either a template retrieval or a classification algorithm can be used. The main objective of these two algorithms is to narrow the search space so that the algorithms of the evaluation phase perform quicker. Either algorithm has pre-selection rules, intended to ensure that the result of the pre-selection complies with given parameters (e.g. number of cases retrieved). In that sense, in order to comply with these rules, the algorithms may run more than once per search, dynamically making adjustments to the search settings.

The template retrieval algorithm basically pre-selects all the cases that may be similar. This can be done as we know a priori which type of cases have the possibility of being similar and which ones do not. As an example, cases that do not address the same norms cannot be similar as they address different legal domains. In that sense, template retrieval works much like SQL queries. Given a case, a set of cases is retrieved from the database with the guarantee that they have certain characteristics (e.g. address at least one of the norms addressed by the new case).

The classification algorithm, on the other hand, uses association rules mined previously to determine to which category a case belongs to. An example rule could be "if party1 uses norms A and B then party1 wins". Thus, the algorithm determines in the first place which category the new case belongs to and then, pre-selects all the cases of that category.

In the evaluation phase, additional operations are performed on the pre-selected cases, such as determining their similarity and ordering them according to it. Similarly, in this phase, two different approaches can be used: one based on a nearest neighbor algorithm and another one based on the cosine similarity concept. The basic idea is that any of these algorithms is only applied to the reduced set of pre-selected cases instead of applying them to the whole case base. Thus, the algorithms start by determining a similarity measure between each pre-selected case and the new case.

In the first approach, the nearest neighbor shown in equation 1 is used.

$$\frac{\sum_{i=1}^{n} W_i * fsim_i(Arg_i^N, Arg_i^R)}{\sum_{i=1}^{n} W_i} \quad (1)$$

In this equation,

- n – number of elements to consider to compute the similarity;
- W_i – weight of element *i* in the overall similarity;
- Fsim – similarity function for element *i*;
- Arg – arguments for the similarity function representing the values of the element *i* for the new case and the retrieved case, respectively *N* and *R*.

In this algorithm, the weights are at this moment determined by a law expert, based on the importance that according to his experience each of the components of the similarity measure has. However, in the future, we intend to let the system change these values dynamically, looking at past iterations, in an attempt to select the most appropriate weights for each case.

We now detail the information of the case that is considered to be relevant for the computation of the similarity with the nearest neighbor approach, i.e., the components. According to our scope of application, we consider three types of information: the objectives of each party, the norms addressed by each party and by the eventual witnesses and the date of the dispute. Both the norms addressed and the objectives are lists of elements. The similarity function consists in comparing two lists (equation 2). The similarity is higher when the two lists have a higher percentage of common members. As for the date, the similarity function verifies if the two dates are within a given time range, the similarity is higher when the two dates are closer.

$$fsim_{list} = \frac{|L_N \cap L_R|}{n}, n = \begin{cases} |L_N|, & |L_N| \geq |L_R| \\ |L_R|, & |L_N| < |L_R| \end{cases} \quad (2)$$

In the second approach, a similarity measure based on the concept of cosine similarity is used [53]. This method uses the binary vector representation of the information and is based on the notion that the similarity between two vectors can be determined by finding the cosine of the angle between them. Given two vectors of attributes *A* and *B*, with *N* entries each, the cosine similarity, θ, is determined as shown in equation 3. The resulting value of similarity of this equation ranges from 0 to 1, with 0 being the smaller value of similarity and 1 meaning that they have exactly the same information.

$$sim = \frac{A \,.\, B}{||A||\,||B||} = \frac{\sum_{i=1}^{N} A_i * B_i}{\sqrt[2]{\sum_{i=1}^{N} (A_i)^2} * \sqrt[2]{\sum_{i=1}^{N} (B_i)^2}} \quad (3)$$

This second method of computing similarity is quite simpler and faster as it uses the vectors of binary entries. However, contrary to the previous similarity function, it does now allow to assign weights to the several components of the case. This may or may not be a disadvantage, depending on the scope of application.

In the evaluation phase another operation takes place which is independent of the method used to compute the similarity: the determination of the utility of each case. This value of utility quantifies how much each case is desirable for each party. After having access to these values, a party can thus compare the retrieved cases in terms of their similarity and their utility and make a personal evaluation of potential advantages versus potential likeliness. The process of computing the utility is relatively simple. Basically, as stated before, each stored case has a solution and an outcome. The outcome denotes the result of applying the solution in the context of that case. The application of a solution to different problems generates different outcomes. The key idea is to apply the solution of each retrieved case to the current problem and determine its value of utility. As a simple example, one could have in a past case:

$$Outcome = worker\ receives\ indemnity\ of\ 2500$$
$$Solution\ = antiquity * 25 * d\_wage$$

This example denotes a case in which a worker received an indemnity of 2500, that was determined by the formula depicted in the solution, defined by Decree of Law (DL) 7/2009 (Portuguese laws). As, in that case, the worker had an antiquity of 5 years and a daily wage of 20, the value of the outcome was obtained applying the formula with these values. Now, let us consider that this case was retrieved because it was similar to the new case and that, in this new case, the worker has an antiquity of 15 years and a daily wage of 17. For this worker, the application of the same solution would mean an outcome of 6375. This adds the last information that is compiled in this phase.

Independently of the method used, the output of this phase is a list of similar cases ordered by their degree of similarity. This information can then be used by the disputing parties, neutrals or by the platform itself in a wide range of applications. Figure 5 shows the textual output of a typical execution of the retrieve process, combining the template retrieval with the nearest neighbor approaches. This figure highlights the process of refining the pre-selection with successive iterations and changes in the search parameters. Figure 6 depicts the interactions that took place between the several agents in this specific example.

All the retrieve process is coordinated by the Coordinator agent, which has the knowledge to interact with the remaining agents, request the necessary tasks and handle the results. Hence, when a new retrieve request arrives to the Coordinator (typically coming from a user interface or another agent), it starts a new Complex Action (CA). A CA is an action that will use more than one Simple Action. In the case of the retrieve process, the CA includes the following SAs: (1) Pre-Selection, handled by the TemplateRetrieval agent; (2) Database Query, handled by the Database agent; (3) Computation of Similarities, handled by the NearestNeighbour agent; (4) Computation of Utilities, handled by the Utilities agent; and (5) Return Results, handled by the Coordinator agent. Note that some of these actions may repeat and that some must be sequential while others may be executed in parallel (e.g. computation of utility and similarity). So, in this case, the Coordinator starts a new case retrieval by requesting a pre-selection of cases from the TemplateRetrieval. In this request, the Coordinator includes only a single rule: to start the pre-selection by considering the article of each norm. The TemplateRetrieval then formulates a database request containing a SQL sentence and sends it to the Database agent, which returns the result. The TemplateRetrieval analyses the result from the Database and determines that the rule that establishes the maximum amount of cases is being violated. Thus, it reformulates the request to the Database in order to get fewer cases. This happens once again until it receives a pre-selection of cases that is in accordance with the rules. At this point, the TemplateRetrieval returns the pre-selection of cases to the Coordinator, which will request the computation of the similarity and the utility values of each pre-selected case to the corresponding agents. In this context, the similarity denotes the extent to which a retrieved case is similar to the current case and the utility quantifies how much the user would win or lose if the solution of the given case

would be applied to his case. When these values are returned, the Coordinator is finally able to compile the result of the case retrieval task and return it to the requesting entity.

```
New Pre-Selection task arrived: < 1291886422498, 1286439756546 >
*****TemplateRetrieval: Requesting for cases that match the rules: (
        depth: article
)
2010/12/09 09:20:23 : Connected to Database
*****Database: Received request to query database from TemplateRetrieval@TIARAC1:1099/JADE
2010/12/09 09:20:23 : Connected to Database
*****Database: Returning query result...
*****TemplateRetrieval: Received a pre-selection of 77 cases
*****TemplateRetrieval: Rule violated: "Max cases". Deciding new pre-select rules...
*****TemplateRetrieval: Changed search depth to: number
*****TemplateRetrieval: Requesting for cases that match the rules: (
        depth: number
)
*****Database: Received request to query database from TemplateRetrieval@TIARAC1:1099/JADE
*****Database: Returning query result...
*****TemplateRetrieval: Received a pre-selection of 77 cases
*****TemplateRetrieval: Rule violated: "Max cases". Deciding new pre-select rules...
*****TemplateRetrieval: Changed search depth to: item
*****TemplateRetrieval: Requesting for cases that match the rules: (
        depth: item
)
*****Database: Received request to query database from TemplateRetrieval@TIARAC1:1099/JADE
*****Database: Returning query result...
*****TemplateRetrieval: Received a pre-selection of 28 cases
*****TemplateRetrieval: SUCCESS: Returning results to agent( agent-identifier :name Coordi
*****Coordinator: Received results from pre-selection of task: 1291886422498
*****Coordinator: Requesting computation of similarities for task 1291886422498
*****NearestNeighbour: Received request for similarities: 1291886422498
*****NearestNeighbour: Sending list of computed similarities to: ( agent-identifier :name
*****Coordinator: Received results from computing similarity of task: 1291886422498
*****Coordinator: Returning result of Get Similarity Request
*****Coordinator: Ending sub task 1291886422498
*****Coordinator: Updating task 1291886422370
*****Coordinator: Starting a new Utilities task: 1291886425356
*****Coordinator: Requesting computation of utilities for task 1291886425356
*****Utility: Received Compute Utility request from ( agent-identifier :name Coordinator@T
*****Utility: RECEIVED REQUEST FOR 28
*****Utility: Sending results from Utility request to ( agent-identifier :name Coordinator
*****Coordinator: Received results from computing utilities of task: 1291886425356
```

Figure 5. The output of the execution of a typical case retrieval process.

Reuse

In the Reuse (or adapt) phase, solutions are adapted to match the characteristics of the new case. In this work solutions are, as stated before, lists of steps that the parties took in order to achieve the verified outcome (typically trade-offs) in a previous case. This is structured information that can thus easily be changed. Therefore, in order to adapt a case that requires to do so, the system starts by looking at the information of its solution. It then searches for differences with the new case. It is in these different points that the adaptation will take place. Several actions can be performed, namely replacing basic information (e.g. names, addresses, dates, places), omitting steps that do not apply in the context of the new case or adding new steps that are needed. The key idea is to change the cases retrieved so that they can be understood by the disputing parties, under their context.

Revise

In this phase, a solution and the corresponding justification is presented to the parties, after which their behavior is analyzed. In order to do so, the cases and their likeliness to occur, according to the characteristics of the current case, are presented to the users in a graphical fashion (Figure 7). In this graphical representation, the parties can see the several cases that were retrieved in the form of the small colored circles. By clicking in these circles, an interface

pops up that shows details about the corresponding case and about the reasons for its selection. These cases span the space between the BATNA and WATNA of each party, so that they can be compared in terms of their utility for each party. In order to determine the values of the BATNA and WATNA, the system uses logic rules defined after the Portuguese labor law. These are rather simple rules that establish the values of eventual indemnities and other parameters of the outcome, based on concepts like worker antiquity, work hours, extra hours, night work, among others.

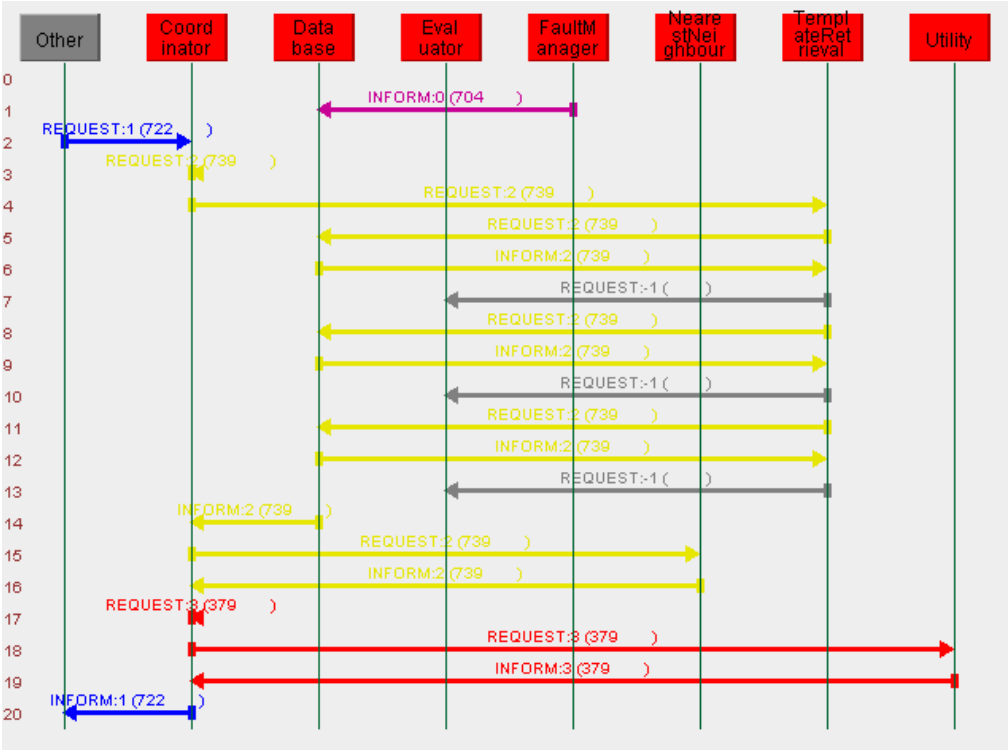


Figure 6. Visualization of a typical communication between agents for retrieving similar cases.

Let us take, as an example, the Portuguese labor law domain, as depicted in Decree of Law (DL) 7/2009 (Portuguese laws), considering a scenario in which a worker wants to end the labor contract claiming that the employer did not pay the last three salaries. According to Article 394th, nr. 2 a), the lack of regular payment of the salary constitutes a just cause for a worker to end the contract. Moreover, Article 394th, nr. 1 states that when there is a just cause, the worker can immediately end the labor contract. The first question is thus to determine the existence or not of the lack of payment, and thus, of a just cause for ending the contract. Assuming that this has been proved, let us try to determine the best and worst scenarios, from the point of view of the worker. The most important norms are found in Article 396th, numbers 1, 3 and 4. Number 1 states that, if Article 394th is true (there is just cause for ending contract), the worker is entitled to 15 to 45 days of salary plus indemnity for each year of contract. It also states that this value varies according to the degree of wrongfulness of the employer and that the total indemnity paid to the worker should not be inferior to three salaries plus indemnity. However, number 3 states that the indemnity paid can be higher whenever the worker suffered property damage or other damage, of higher value. Finally, number 4 states that, in the cases of a temporary employment contract, the value of the indemnity cannot be smaller than the value of the salaries that would be received until the end of the contract. Thus, the resemblance between legal norms and rules, allows formalizing the computation of the BATNA and WATNA in the form of IF-THEN rules. The example addressed here is presented below.

A simplification of the rules that allow the computation of the BATNA and WATNA values according to the Portuguese labour law. This example code considers only the case in which a worker ends the contract with a just cause. M_SALARY denotes the monthly salary; D_SALARY denotes the daily salary; M_REMAINING denotes the months remaining until the end of the temporary contract; +VARIABLE denotes an unknown value, higher than VARIABLE.

```
Def_Rule 396
if RULE_394 then
  WATNA := 3 * (M_SALARY + SENIORITY)
  if TEMPORARY_CONTRACT then
    if WATNA < M_REMAINING *(M_SALARY + SENIORITY) then
      WATNA := M_REMAINING *(M_SALARY + SENIORITY)
  if WATNA < 15 * (D_SALARY + SENIORITY) then
    WATNA := 15 * (D_SALARY + SENIORITY)
  BATNA := 45 * (D_SALARY + SENIORITY)
  if BATNA < DAMAGE then
    BATNA := +DAMAGE
```

However, in Figure 7, besides looking at the retrieved cases and the BATNA and WATNA values, a disputing party can analyze additional information. Clusters are created that are intended to show some grouping between the cases, allowing a first analysis from a higher point of view. The cases belonging to the same cluster are shown depicted with the same color. The interface also shows, for each cluster, the mean values of similarity and utility. A linear regression is also drawn, which may help the disputing party to determine in which region an outcome is more likely or more valuable, according the utility. This regression is also used to depict the MLATNA: the region in which the line of the regression is green instead of black denotes the most likely region for the outcome. Finally, using this interface it is also possible to delimit the ZOPA, or the Zone of Potential Agreement. This concept was proposed by [3] and denotes the intersection of the range of possible outcomes of the parties, i.e., what can happen.

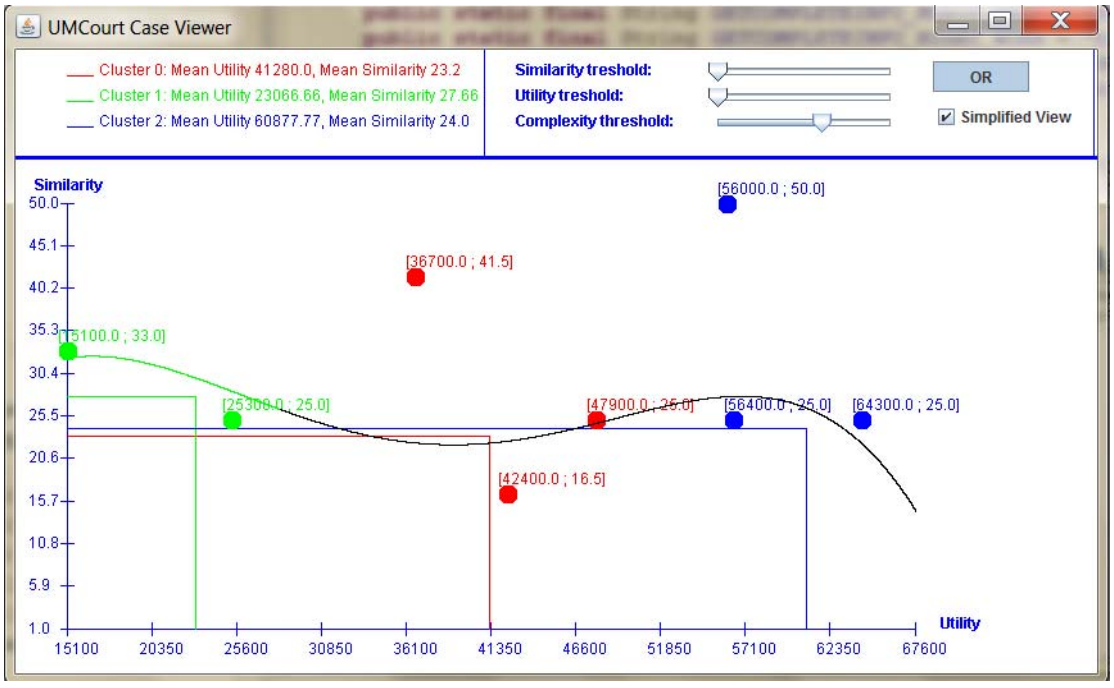


Figure 7. The prototype of the interface showing the user the compiled information. This interface shows several cases retrieved from the case-base, displayed on a Cartesian plane. Only the cases more similar to the target case are selected. The horizontal axis represents the utility while the vertical one represents the similarity. Each colored dot represents a case and different colors indicate that cases belong to different clusters. Cases can be clicked on to access further information.

Retain

This CBR process can be used as it was presented until now, as a tool to compile useful information, or it can be integrated in a higher level process (Figure 8). Thus, at the end of the last phase, one of several things may happen. The parties may choose to accept the proposed outcome by committing to implement the steps in the solution. In this case, the system increases the value of the case proposed, indicating its successful application to solve a case with these specific characteristics. Alternatively, the parties may not agree to accept the proposed solution. In this case, the parties can make use of a negotiation module that allows them to search for alternative solutions by modifying proposed solutions in order to maximize the hypothesis of agreement. Moreover, the parties can decide to stop the alternative resolution process and either advance to a court or chose another dispute resolution method. In this case the process is considered to have failed.

If they decide not to go to a court (e.g. by choosing other methods for conflict resolution), the value of the case proposed is decreased, denoting that it was not applied successfully. If this occurs, the process ends. In order to gather more information that can help to improve the system, the parties may be asked to complete a questionnaire stating why, in their opinions, the process failed. This information will later be useful for improving the system by changes in the process model, in the way information is provided, among others. Alternatively, if the parties decide to go to court, the process continues by waiting on the decision of the court. If the decision concurs with the solution that was proposed by the system, the value of the proposed case is increased and the new case that resulted from the court is added to the case base. This, in fact, constitutes another innovation of the UMCourt platform, as the only information needed to complete the data of the case is the outcome. All the remaining information has already been provided by the parties when using the system. This makes the addition of cases a simple and fairly automated process. If, however, the outcome of the trial differs from the proposed solution, the system will decrease the value of the proposed case and will try to find reasons for the failure of the CBR process. In this situation as well, a new case is added to the case base.

In the case in which the parties want to work the solution of the dispute, using as a starting point the proposed solution, they can use the negotiation or the mediation module embedded in the platform. This is described in detail in the section 5.

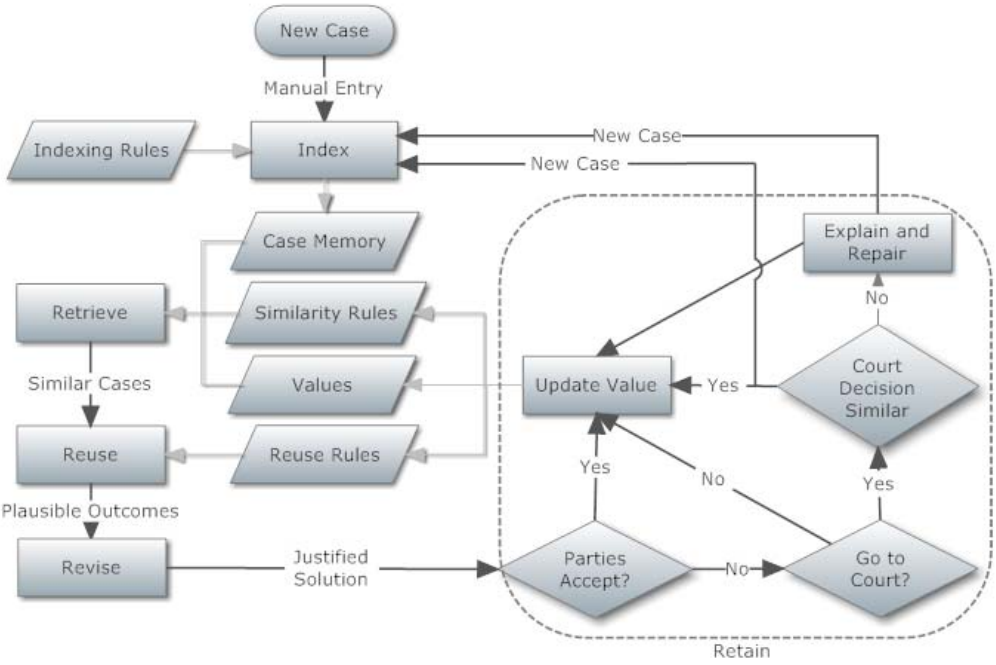


Figure 8. A Flowchart depicting the integration of the CBR model presented in a higher level complete process of dispute resolution. Rectangles represent processes, parallelograms represent information structures and rhombuses represent decisions.

5. Using Negotiation and Mediation to Improve CBR Efficiency

As mentioned above, the CBR process may fail, namely if the parts do not agree on the outcome proposed. In such scenarios, negotiation or mediation are additional paths that can be followed to search for alternative solutions. This requires the active participation of the parties. These two approaches have been implemented, supported by the same CBR process presented, following a blackboard approach. Therefore, there is a shared space to which the parties publish and debate proposals. This is implemented by the Blackboard agent, which is able to receive and interpret messages from the parties, and take decisions according to the content of those messages in order to guide the process to a successful conclusion (Figure 9).

Setting up the Process

The mediation and negotiation processes build partially on the previously described functionalities. In that sense, before the actual start of the process, some information is generated that will help define the whole process. This information includes the already mentioned BATNA and WATNA. These values are particularly interesting during a negotiation or mediation as, besides helping parties have a clear picture of the possible outcomes, they might also be used as a way to put pressure on the other party, especially in dispute resolution procedures that allow the choice of going to court [10].

The remaining values mentioned before (e.g. ZOPA, MLATNA) as well as the ordered list of similar cases will also be used as described ahead. Once all this information is compiled, parties may register on the platform by means of their personal agents and the mediation process is ready to start. How this process takes part is detailed below.

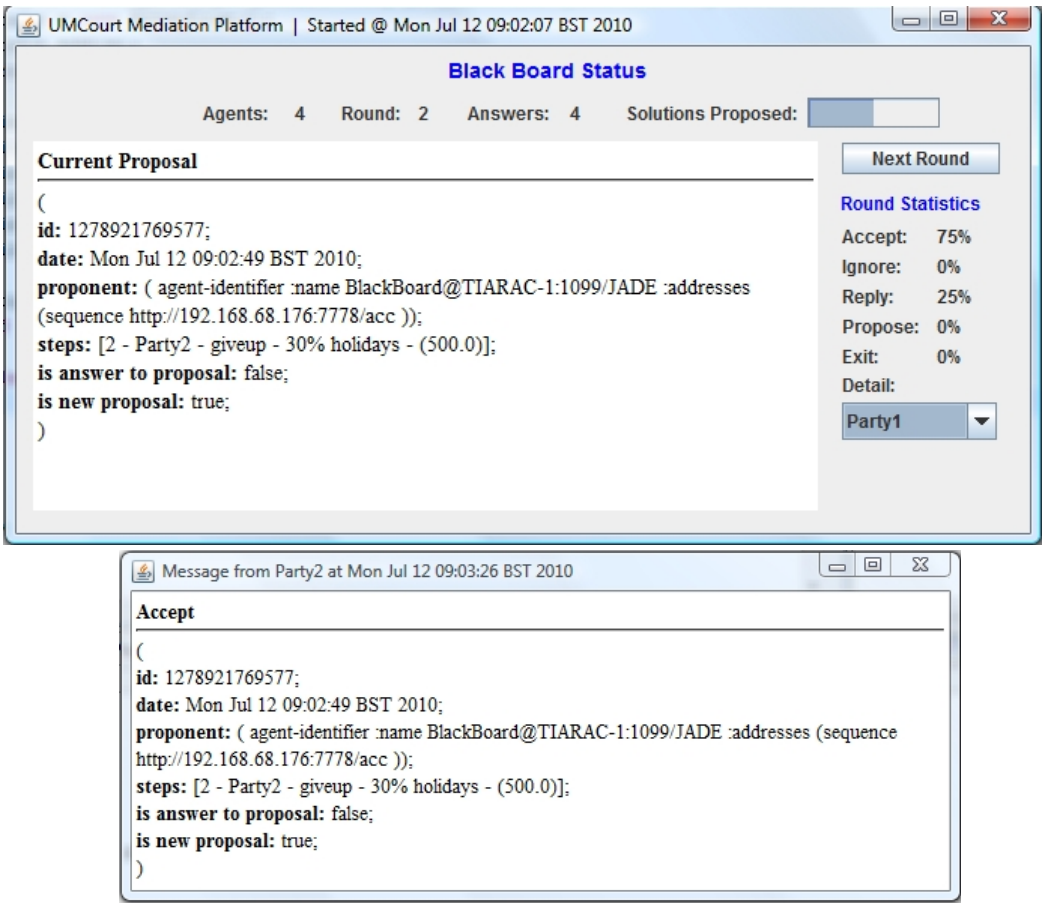


Figure 9. A simple visual interface for showing the black board status (upper figure) and the content of a reply (lower figure).

5.1. Mediation

The objective of the mediation process is to suggest alternative solutions to the parties in order to overcome impasses in scenarios in which the parties cannot agree on a given solution. Moreover, each solution that is proposed can be changed by the parties in an iterative process that allows them to draw the most mutually satisfactory solution. This means that this approach is suited for cases in which the two disputant parties have a behaviour in which they effectively propose solutions and want to actively participate on the definition of the outcome. Specifically, parties that have a collaborating or compromising conflict style [46] should use this approach. All this process is managed by an electronic mediator represented by the blackboard: it receives messages from all the parties, does the necessary computation and publishes a new proposal to be analysed. These messages may include proposing new solutions, refusing or agreeing with the current proposal, ignoring a proposal, among others. Furthermore, the blackboard has a list of solutions taken from past known cases, ordered in decreasing order of similarity, i.e., the MLATNA is at the head of the list.

After the parties register, the process starts with the blackboard publishing the first proposal for solution, which is the solution given by the MLATNA. Whenever a new proposal is published by the blackboard each party can perform one of several actions. The party can accept the proposal denoting that it agrees with it or, otherwise, it can refuse it. If the party wants to propose a new solution that consists on the modification of the present one, it can make the intended changes to the current proposal. Alternatively, the party may also suggest a completely new solution. Finally, one party may also choose to ignore the current proposal or it may leave the mediation process, ending it.

This iterative process is entirely controlled by the blackboard agent and is organized into sequential rounds (Figure 10). Each round starts with the blackboard publishing a new proposal. This proposal may have several origins. It can come from the list of similar cases or it may be computed from the previous answers of the parties. In this case, if only one party answered with either a new proposal or a modified proposal, that will be the next solution to be proposed by the blackboard. Otherwise, the blackboard will search for similarities between the several proposals of the parties and compute a new solution based on those similarities. If, however, the proposals are divergent, the blackboard will propose the following solution from the list of similar cases, if any. This process ends when all the agents send an agree message or at least one sends an exit message. Moreover, the process also ends when the parties do not agree on a solution before the system suggesting all the solutions from the list of similar cases, that is, the system runs out of known solutions and the parties did not agree on any of the suggested ones.

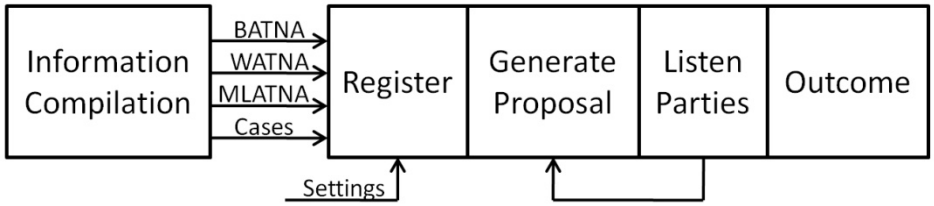


Figure 10. The several steps of the mediation process model.

The description of the algorithm is presented below.

```
Algorithm Mediation is
  input:  List of cases with solutions, C
          List of parties, P
  output: A solution for the dispute

  round := 0      (identifies the round)
  msgSet := []    (a list of received messages)
  agree := 0      (the number of agents that agree)
  exit := 0       (the number of agents that exit)
  proposal := []  (the current proposal)
```

```
i := 0          (the case C being currently proposed)

while (agree < length(L) and exit < 1)
    if (round = 0)
        proposal := Ci
    else if (msgSet contains "ReplyTo" or msgSet contains "Propose")
             set := ReplyTo in msgSet ∪ Propose in msgSet
             proposal := intersect set
         else i++
              if (i = Length(C))
                  return null
              else proposal := Ci
    publish proposal
    msgSet := []
    for each party in L
        msg := receive from party or timeout
        msgSet := msgSet ∪ msg
    agree = count "agree" in msgSet
    exit  = count "exit" in msgSet
if (exit > 0)
    return null
else
    return proposal
```

Algorithm 1. The description of the Mediation algorithm.

5.2. Guided Negotiation

This process has a similar objective to the mediation process but tries to achieve it in a different fashion. It has been developed having in mind conflict scenarios in which the parties exhibit essentially an avoiding conflict style [46], i.e., one or more parties are reluctant or have difficulties in generating possible solutions. In that sense, the responsibility of generating solutions falls entirely on the system to which the parties can answer in three ways: either they ignore, agree or do not agree with the current proposal.

The negotiation process develops around a list of previous negotiation processes provided by the CBR mechanism that will be used to guide the parties through their current process (Figure 11). Each negotiation process is defined by a list of events or steps in which each one represents a sequential stage in the negotiation process.

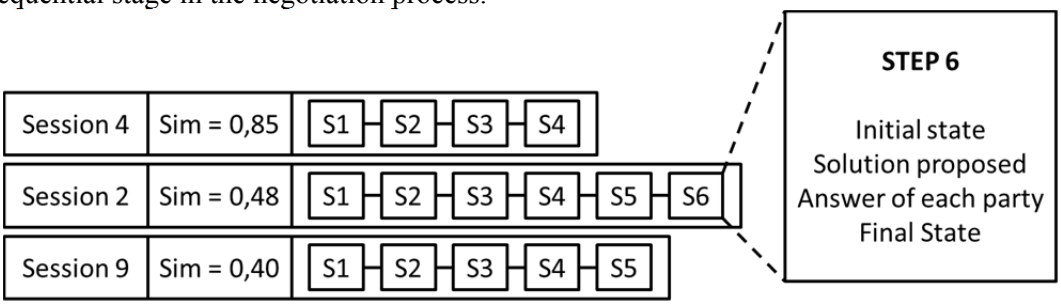


Figure 11. Representation of the information needed for guiding a negotiation process, consisting of several sessions with respective value of similarity.

Each step contains the initial state (denoting the initial conditions), a proposed solution (a list of actions or steps that one or more parties must perform, usually trade-offs), the answer of each party to the proposed solution and the final state (denoting the final conditions). The process starts with the blackboard publishing the first negotiation step of the most similar negotiation session, which is determined using a similarity function from the described CBR algorithm. At this point, parties reply to the published proposal. When answering positively, the parties will be stating that they agree to implement the steps that the negotiation step describes and that they

accept the corresponding outcome. When answering negatively, they state that they agree neither with the proposed solution nor with the outcome.

At the end of each round, i.e., when all the messages from the parties are received by the blackboard, the content of these messages is analysed. At this point, the negotiation can follow three distinct paths. If all the agents agreed on the current proposal, the negotiation process successfully terminates as a solution that satisfies all parties has been achieved. Otherwise, if the majority of the agents agree with the current proposal, the blackboard will propose the next step on the same negotiation process on the following round. Alternatively, if the majority of the agents does not agree with the current event, the blackboard will fall back and select the following negotiation process on the list and start again from its first step. This denotes that the majority of the parties do not agree with the current negotiation path and a new one will be selected. This process repeats until either all the parties agree on a solution, the system reaches the last step of the last negotiation process in the list or the majority of the agents do not agree on an event of the last negotiation process. The whole negotiation process is depicted in Figure 12 and in the algorithm described below.

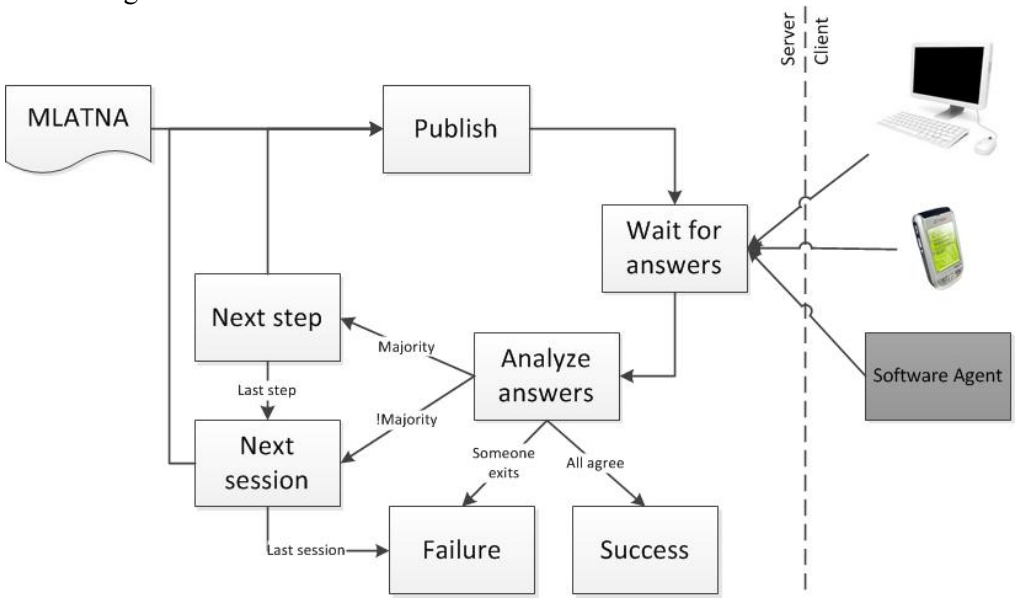


Figure 12. The process of guiding a negotiation coordinated by the blackboard.

```
Algorithm Mediation is
  input:  List of previous mediation processes, L
          List of parties, P
  output: A solution for the dispute

  round := 0      (identifies the round)
  msgSet := []    (a list of received messages)
  agree := 0      (the number of agents that agree)
  exit := 0       (the number of agents that exit)
  proposal := []  (the current proposal)
  i := 0     (the mediation process being currently used)
  j := 0          (the current step being proposed)

  while (agree < length(L) and exit < 1)
       mediation := L_i
       solution := mediation_j
       publish proposal
       for each party in P
           msg := receive from party or timeout
           msgSet := msgSet U msg
       agree = count "agree" in msgSet
       exit  = count "exit" in msgSet
```

```
if (exit > 0)
    return null
if (agree = Length(P))
    return solution
else if (agree > Length(P)/2)
          j++
     else i++
          j := 0
if (j = Length(mediation))
     j := 0
     i++
if (i = Length(L))
     return null
round++
```

Algorithm 2. The description of the Guided Negotiation algorithm.

6. Results

Let us now describe the main results of the work presented in this paper. UMCourt is being tested and assessed in a setting with the characteristics that follow. Given the complexity of the Portuguese Labour law, for test purposes we are dealing with a restrict set of norms. In that sense, we are considering a group of 36 norms of the Decree of Law 7/2009, from February 12th, 2009. These norms were selected because they are generally present in most of the labour-related disputes and address the following domains: (1) functions performed by the employee; (2) effects of the lack of professional title; (3) employee's rights; (4) employee's obligations and (5) general obligations of the parties. The database used in these tests contains a total of 127 indexed cases (Figure 13). The representation of these cases in vectors of binary entries was generated previously by the system and was also made available, in the form of a file.

```
Statistics of the case base:

Article 118, Funções desempenhadas pelo trabalhador, 6%
Article 117, Efeitos de falta de título profissional, 4%
Article 129, Garantias do trabalhado, 38%
Article 128, Deveres do trabalhador, 44%
Article 126, Deveres gerais das partes, 6%


Total cases in the database: 127
```

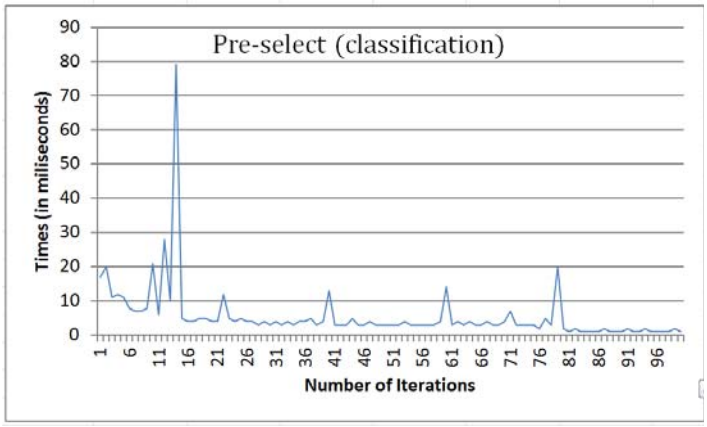
Figure 13. A description of the database detailing which aspects of the Portuguese labor law are addressed by the cases and in which proportion. Article 118 concerns functions performed by the employee; Article 117 is about the effects of the lack of professional title; Article 129 concerns employee's rights; Article 128 depicts employee's obligations; and Article 126 concerns the general obligations of the parties

6.1 Efficiency

During this process, we have mainly focused on collecting data about two main subjects: efficiency of the different algorithms and platform self-assessment. Hence, concerning the first subject, multiple iterations of the algorithms were executed, under different settings, and their times of execution measured. This will be useful to compare them in terms of their execution times. These tests thus evaluate the efficiency of each algorithm presented. We acknowledge that the most important factor in the efficiency of the algorithms is the size of the case base. As depicted in Figure 13, the database contains at the moment 127 cases decided in court, provided by legal experts. The more cases there are the higher the efficacy of the algorithms. However, efficiency will necessarily decrease. Nevertheless, the algorithms presented here have linear

complexity. In that sense the efficiency of these algorithms will decrease proportionally to the size of the database.

Pre-select (Classification)

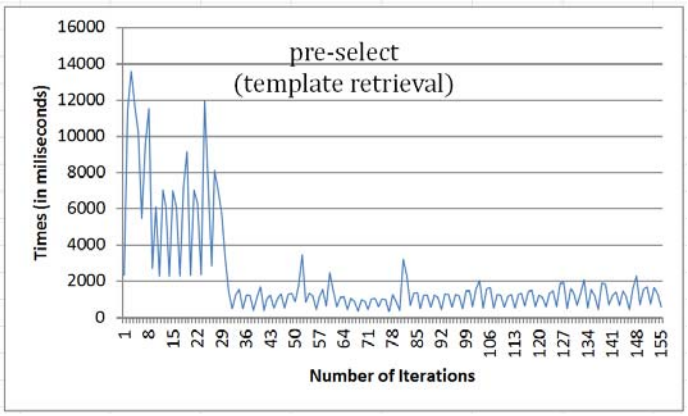


Mean	5,53 ms
Maximum	79 ms
Minimum	1 ms
Iterations	100

Figure 14. Results for the classification algorithm of the pre-selection phase.

The data depicted in figure 14 concerns a total of 100 iterations of the pre-selection algorithm that uses the association rules to retrieve cases according to their classes. To test it, in each iteration a random case was provided to the algorithm. The algorithm thus had to analyze the cases, determine which rules applied in the cases and then pre-select, from the file containing the vectors of all the cases, the ones that belonged to the same category of the case provided. Analyzing figure 14, it is possible to conclude that this is a considerably fast process. This is essentially due to the fact that: (1) the representation of the data as vectors had been previously performed and (2) cases are already indexed according to the rules they comply with. Therefore, once all this information is made available, this is a highly efficient algorithm for information retrieval. Another factor contributing to this is that the data is stored locally. This is possible given the small size of the data when sored according to the Vector Space model, constituting another advantage of this method.

Pre-select (Template Retrieval)



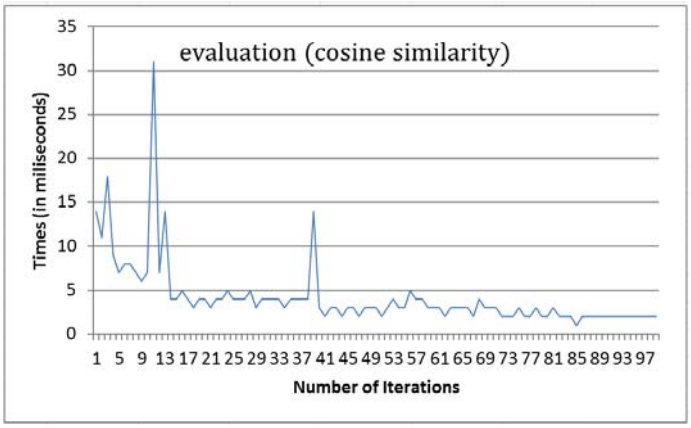
Mean	2203,8 ms
Maximum	13607 ms
Minimum	313 ms
Iterations	155

Figure 15. Results for the template retrieval algorithm of the pre-selection phase.

To test this algorithm, several pre-selections were requested, using a random case. To perform the pre-selection, the algorithm must analyse the case and then interact with the database, requesting all the cases that match a given criteria. This algorithm was tested using a local instance of a database and a remote one. Looking at Figure 15, the first 30 values correspond to the tests in which a remote instance was used while the remaining correspond to the use of the local instance. There is a visible difference between the two scenarios. This difference is

aggravated by the fact that: (1) the database is not a dedicated one, (2) being distributed, the system depends on external factors like the velocity of the internet connection and (3) in order to satisfy the pre-selection rules, the algorithm may need to make several iterations (requests to the database). In this sense, one may improve the algorithm by choosing the best pre-selection rules, minimizing the number of requests to the database. This is addressed further ahead. When compared with the previous algorithm, this shows significantly higher times of execution, mainly due to the interaction with a remote database.

Evaluation (Cosine Similarity)

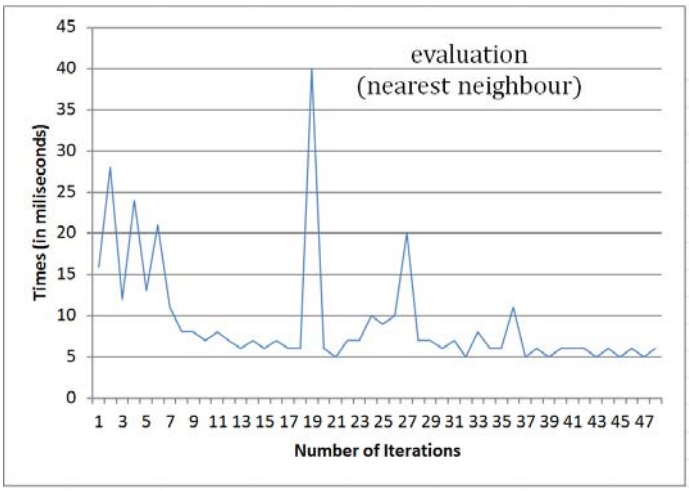


Mean	4,2 ms
Maximum	31 ms
Minimum	1 ms
Iterations	100

Figure 16. Results for the Cosine similarity algorithm of the evaluation phase.

This algorithm uses the cosine similarity formula described before to determine the similarity between two cases. To test it, the algorithm was provided a case and a list of cases with the objective of determining the similarity of each case in the list with the isolated case. The results depicted in figure 16 show that this is a relatively fast way of computing the similarity, mainly because it only deals with binary values. However, a major disadvantage of this algorithm is that it does not allow assigning weights to the different components of a case.

Evaluation (Nearest Neighbour)



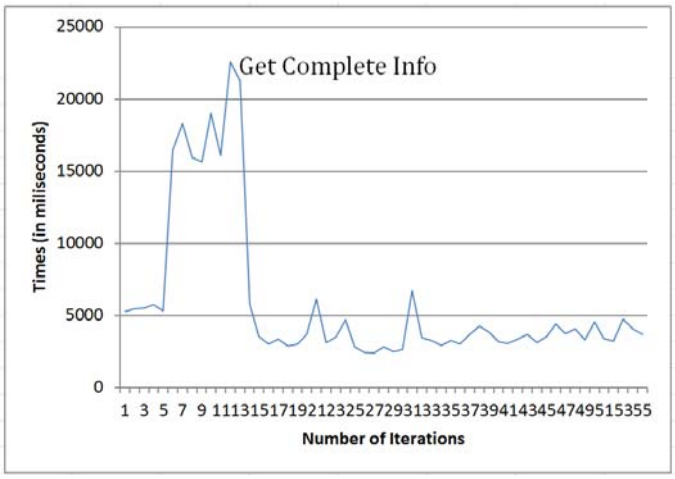
Mean	9,3 ms
Maximum	40 ms
Minimum	5 ms
Iterations	48

Figure 17. Results for the nearest neighbor algorithm of the evaluation phase.

To test this algorithm, we have proceeded similarly to the previous one. Looking at the collected data depicted in figure 17, it is possible to conclude that the nearest neighbour algorithm performs slightly slower. As both algorithms have linear complexity, this poorer performance can be attributed to the fact that this algorithm deals with several types of variables (e.g. integers, strings, floating points) rather than binary ones. However, this algorithm allows for

weights to be assigned to the different components of the similarity function, allowing an evaluation that might be closer to the one performed by a human expert.

Get Complete Info



Mean	5891,6 ms
Maximum	22580 ms
Minimum	2394 ms
Iterations	55

Figure 18. Results for the process that compiles a complete set of information for a given dispute.

In this test, the objective was to determine the efficiency of a request of all the information regarding a case, i.e., giving a random case to the platform, how much time does it take to compile all the possible information for the user. This includes, as described before, pre-selecting and evaluating cases, computing the BATNA, WATNA, MLATNA, ZOPA as well as building the visual representation of the information. For this purpose, the algorithms described above were randomly selected to be used. Both local and remote requests were made, which is reflected in the execution times, similarly to the previously presented results. Thus, the execution time of a complete info request depends mostly on which algorithms are selected. The resemblance of the graph depicted in figure 18 with the one depicted in figure 15 is also not a coincidence as, given the potentially high values of its execution times, it has a considerable influence on the overall performance.

6.2. Efficacy

More than the efficiency of the platform, one must also analyse its efficacy, i.e., it is not enough to perform a given task quickly, it must also be done correctly. Thus, regarding the second subject being tested, the platform keeps record of some key actions as well as their results. This allows, in a first instance, to determine which actions fail the most. Then, together with a description of the possible problems, eventual causes and eventual solutions, the platform is able to provide advice about what parameters to change in order to potentially improve its efficiency. Our objective is that, in the long term, the platform applies autonomously these recommendations. However, this still requires further validation as it is a very sensitive topic. A typical output of a self-assessment request is shown in figure 19. In this example, the platform is assessing the performance of the preselect action. First of all, it provides information about the amount of times that the action failed or succeeded. Here, failing means that the pre-selection violates some rule (e.g. regarding the number of cases) and must be reformulated and re-run. On the other hand, succeeding means that the pre-selection respects all the rules. In the example, this action is failing in 66% of the cases, corresponding to a total of 543 cases. Following, the platform points out the top reasons for failure as well as their frequency. In this example, the pre-selection fails mostly because too many cases are being pre-selected. Other minor reasons include not enough cases being pre-selected or reaching a state in which it is not possible to satisfy all rules. This happens when it is not possible to manage the pre-selection settings with enough precision or when the pre-selection rules are too strict. Finally, given this, the platform points out several possible actions that might be used to address the described problems. In this

case, three actions are suggested: (1) changing the rule that establishes the maximum amount of cases that should be pre-selected (this would actually decrease the number of errors but might not be good for who deals with all the cases later); (2) changing the initial search depth (this is more advisable as changing the initial search parameters might lead to a better result faster) and (3) changing the rule that establishes the minimum cases that should be selected.

```
*****Evaluator: Indexed cases in the database: 127
*****Evaluator: Evaluation of System Performance:
*****Evaluator: Action: preselect
*****Evaluator: Successfull iterations: 33% (271)
*****Evaluator: Failed iterations: 66% (543)
*****Evaluator: Top reasons for failure
*****Evaluator: Code: 1; Description: Too many cases were pre-selected ; 61% (501)
*****Evaluator: Code: 0; Description: Not enough cases were pre-selected ; 2% (23)
*****Evaluator: Code: 2; Description: Not possible to satisfy all rules ; 2% (19)
*****Evaluator: Recommended actions:
*****Evaluator: Change Max Cases rule (33%)
*****Evaluator: Change initial search depth (33%)
*****Evaluator: Change Min Cases rule (33%)
```

Figure 19. The output of a self-assessment request for the pre-select action.

7. Conclusions and Lessons Learned

In this paper we have presented the underlying architecture of UMCourt as well as its main functionalities, after having described the legal domain being addressed. Throughout the work, we placed a special emphasis one the challenge of determining the possible outcomes for the cases, based on the observation of past cases, having as motivation the concept of legal precedent. This task is supported by a CBR algorithm which was also described. The main advantage of this algorithm lies, from our point of view, in the fact that it can be used as it was presented (a tool to compile useful information for the parties, the neutral or the platform itself) or it can be integrated in a higher level process which includes the parties going into litigation or choosing another dispute resolution method, such as mediation or negotiation.

The CBR algorithm was detailed in terms of its several phases. During the description of each of these phases, we have also highlighted another key point of this work: several techniques can be combined to implement the algorithm. This will lead to different possibilities and different results. It is thus up to who uses the algorithm to decide, depending on the type of information needed or the domain of application, which technique to use.

Another key point presented is the close integration between the CBR algorithm and the negotiation and mediation processes. In fact, it is this same algorithm that provides all the information that is necessary to inform the parties and for the platform to act as a neutral and guide these processes. Moreover, human neutrals can also make use of the information compiled in order to get knowledge about past similar cases so that more rational decisions can be made. And, as pointed out before, this can be useful in both common and civil law systems.

On the use of Case-based Reasoning in the Legal Context

The use of case-based reasoning for conflict resolution is a natural approach in common law contexts, although it's use in civil law systems also makes sense, as described above. Moreover, negotiators and mediators in general rely on their past experiences in order to take better decisions. These were in fact the reasons that supported our decision to follow a case-based approach. Traditionally, two main drawbacks are associated to case-based reasoning [30].

On the one hand, case-based approaches may suffer from inefficiency, mostly when the size of the database grows. Concerning this subject, it was our decision to develop several methods so that their performances could be tested in order to determine the more efficient ones. Moreover, we also acknowledge that conflict resolution is generally an asynchronous process, which takes place over several hours or even days. Concerning this subject, we believe to have

succeeded as the methods described here have relatively low times of execution and the context of application does not mandatorily demand for higher performance.

On the other hand, the efficiency and efficacy of case-based approaches is also known to depend directly on the number and characteristics of the cases in the database. And this we acknowledge to be the more serious challenge to overcome. The number of cases, as it is evident, influences the performance of the algorithms but also influences its efficacy. Moreover, case-based approaches are highly domain-dependent, i.e., cases take place in a given legal context and are not easily adapted to other domains as the norms are different. In our experiment we used a database whose cases focused mostly on employee's rights and employee's obligations (articles 129 and 128 of the Portuguese labor law, respectively), although some cases also addressed other articles. Our experiment consisted in conflict resolution scenarios in the labor law domain, set up by users, involving the issues addressed by the five articles (and respective numbers and items) addressed by the database (Articles 118, 117, 126, 128 and 129). The users were students and teachers of master courses on Law and on Informatics in our institution. We concluded that disputes involving employee's rights and obligations had many more solutions proposed by the platform than disputes involving other issues, diminishing the success rate of the resolution of the second ones.

In that sense, our main conclusion in this subject is that a pure case-based approach can be quite effective in generating solutions for a conflict resolution in scenarios in which there are enough past cases. In that sense, the domain of the database should be explicitly defined in order to define the domain of the conflict resolution platform. However, a hybrid approach can be used in which case-based reasoning is supported by other tools that can generate solutions when a case-based approach alone is not enough. On the one hand, it is possible to rely on parties themselves to generate solutions, although this approach tends to fail when parties are unable or unwilling to do it. On the other hand, other technological tools can be used for the generation of solutions. Namely, we are now working on genetic algorithms to create solutions in the scenarios in which a case-based approach is not enough to guarantee satisfactory results. The main advantage of this approach is that it depends only on the rules of each specific domain, which are needed to ensure the validity of the solutions. Given that, the algorithm will be able to generate a wide range of solutions from which the most relevant ones will be selected to complement the case-based approach.

On the use of Negotiation/Mediation for Conflict Resolution

The use of negotiation or mediation for conflict resolution is indeed one of the most effective ways of solving disputes out of court. Nevertheless, from our experiments we learned that these processes, by themselves, may not be enough. One of the first issues we had to deal with was a consequence of the fact that most of our users from the informatics field had very little to no knowledge about Portuguese labor law or about conflict resolution at all, as many of the parties involved in conflict resolution do. In that sense, the resolution process often failed because of a clearly unrealistic view of their chances in the dispute. Moreover, these users generally had no idea about the possible outcomes for each side and could not really evaluate how good or bad a given solution was.

Our first conclusion about the use of these alternative methods is that negotiation or mediation alone are not enough. There is, more than anything else, the need for tools that can effectively inform the parties about the possibilities, so that they can take better and more informed decisions. Namely, we found it crucial for parties to know their best and worst possible scenarios, the most likely one as well as some past cases that can be used as learning examples, providing a notion of reality. Our decision was thus to implement this compilation of information before the actual process started, so that the parties could gain a realistic view about their conflict. After implementing this process, we noticed that parties would converge more quickly to a solution that was realistic and in line with the solutions retrieved from the past similar cases. This increased the success rate of the conflict resolution process.

Another issue that was detected was that the success rate of the process depended on the attitude of the parties, i.e., when our users were actively creating solutions and collaborating for

the resolution of the dispute the process was more likely to succeed than when one or more users were not or could not create solutions, limiting their actions to replying affirmatively or negatively to the solutions proposed. In that sense, we developed the two processes depicted in section 5: one targeted at parties that are able and willing to create solutions for the resolution of the conflict and the other for parties that are not. Evidently, the challenge here is still the one of accurately determining the conflict style of each party (e.g. cooperative, collaborative, avoiding). In that sense, we are now developing an automated and non-invasive method that is able to classify the conflict style of a party based on their behavior during the process (e.g. is the party proposing solutions?, are the solutions proposed selfish?, is the party simply answering positively or negatively?). Based on that, the platform will be able to determine the conflict style in real time and will be able to choose the best possible method as well as to adapt during the process.

Concluding, the approaches presented in this work have as main innovations: the enrichment of the information retrieval process with several alternatives, allowing a user (or the platform) to select the one that most suits their needs and a close integration between the information retrieval methods and the posterior processes (e.g. negotiation, mediation, trial) in a single platform in a transparent way. This not only saves resources and time as it increases the organization of the information and makes it easier for parties and practitioners to use it. All this will result in richer conflict resolution processes that, by being supported by information and decision support systems, will lead to better and more mutually satisfactory outcomes.

Acknowledgments.

The work described in this paper is included in TIARAC - Telematics and Artificial Intelligence in Alternative Conflict Resolution Project (PTDC/JUR/71354/2006), which is a research project supported by FCT (Science & Technology Foundation), Portugal. The work of Davide Carneiro is also supported by a doctoral grant by FCT (SFRH / BD / 64890 / 2009).

References

[1] Brown, H. & Marriott, A. (1999). ADR Principles and Practice. Sweet and Maxwell.
[2] Zeleznikow, J., & Bellucci, E. (2003). Family_Winner: integrating game theory and heuristics to provide negotiation support. In Proceedings of Sixteenth International Conference on Legal Knowledge Based System (pp. 21-30).
[3] Raiffa, H. Art and Science of Negotiation. 1982: Harvard University Press.
[4] Lewicki, R., Saunders, D., Minton, J. Zone of Potential Agreement. In Negotiation, 3rd Edition. Burr Ridge, IL: Irwin-McGraw Hill. 1999.
[5] Walton, P.R.E. & McKersie, R.B. (1965). A behavioral theory of labor negotiations. McGraw-Hill.
[6] Bellucci, E., & Zeleznikow, J. (2001). Representations of decision-making support in negotiation. In Journal of decision systems, 10(3-4), pp. 449-479.
[7] Jennings, N., Faratin, P., Lomuscio, A., Parsons, S., Wooldridge, M., & Sierra, C. (2001). Automated Negotiation: Prospects, Methods and Challenges. Group Decision and Negotiation, 10(2), 199-215.
[8] Zeleznikow, J., & Bellucci, E. (2004). Building Negotiation Decision Support Systems by Integrating Game Theory and Heuristics. Artificial intelligence and law.
[9] Fudenberg, D. A. & Tirole, J. (1983). Game Theory (Chapter 1, Section 2.4). MIT Press.
[10] Fernandes AM., Direito de Trabalho, Almedina, 2005 (in Portuguese).
[11] Martinez P. R., Monteiro L., Vasconcelos J., Vialonga J., Brito P., Dray G., Silva L. G., Código do Trabalho, Almedina, 2008 (in Portuguese).
[12] Thiessen, E. M., & Fraser, K. (2003). Mobile ODR with SmartSettle. In Proceedings of the UNECE Forum on ODR.

[13] Ashley, K.D. Case-Based Models of Legal Reasoning in a Civil Law Context. International Congress of Comparative Cultures and Legal Systems of the Instituto de Investigaciones Jurídicas. 2004.
[14] Wooldrige, M. (2002). An Introduction to Multiagent Systems, John Wiley & Sons.
[15] Wooldridge, M. & Jennings, N.R. (1995). Intelligent Agents: theory and practice. In The Knowledge Engineering Review, p. 115-152.
[16] Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems, 3(3), 285-312.
[17] Behrman, B. W. and Davey, S. L. (2001). Eyewitness Identification in Actual Criminal Cases: An Archival Analysis. Law and Human Behaviour, 25(5): 475-491.
[18] Foxall, G. (2004). What judges maximize: towards an economic psychology of the judicial utility function. Liverpool Law Review 25: 177-194.
[19] Posner, R A. (1993). What do Judges and Justices Maximize, Supreme Court Economic Review, 3: 1-41.
[20] Bellucci, E and Zeleznikow, J. (2006). Developing Negotiation Decision Support Systems that support mediators: a case study of the Family Winner system, Journal of Artificial Intelligence and Law, 13(2): 233-271.
[21] Zeleznikow, J. Bellucci, E., Schild, U.J., Mackenzie, G. (2007). Bargaining in the shadow of the law - using utility functions to support legal negotiation. Proceedings of the 11th international conference on Artificial intelligence and law, ACM, pp. 237-246.
[22] Thiessen, E. M. and McMahon, J. P. (2000). Beyond Win-Win in Cyberspace. Ohio State Journal on Dispute Resolution, 15: 643.
[23] Brams, S.J., Kilgour, D.M. (2001). Fallback bargaining, Group Decision and Negotiation, vol. 10, pp. 287–316.
[24] Bellucci E., Lodder A., Zeleznikow J. (2004). Integrating artificial intelligence, argumentation and game theory to develop an online dispute resolution environment, ICTAI-2004 - 16th IEEE International Conference on Tools with AI, pp. 749-754.
[25] Popple, J. (1996). A pragmatic legal expert system, Dartmouth, Ashgate.
[26] Zeleznikow, J., Stranieri, A. (1995). The split-up system: integrating neural networks and rule-based reasoning in the legal domain. Proceedings of the 5th international conference on Artificial intelligence and law, College Park, Maryland, United States: ACM, pp. 185-194.
[27] Landes, W.M., Posner, R.A. (1976) Legal precedent: A theoretical and empirical analysis, Journal of Law & Economics, vol. 19.
[28] Zweigert, K., Kötz, H. (1998). An Introduction to Comparative Law, Clarendon Press, 3rd edition.
[29] I.N.C.M. (2009). Código do Trabalho (in Portuguese).
[30] Aamodt, A., Plaza, E. (1994) Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications, 7(1):39-59, IOS Press.
[31] Goldberg, S.B., Sander, F.E., Rogers, N., Cole, S.R. (2003). Dispute Resolution: negotiation, mediation and other processes, Aspen Publishers, New York.
[32] Notini, J. Effective Alternatives Analysis in Mediation: "BATNA/WATNA" Analysis Demystified, (http://www.mediate.com/articles/notini1.cfm). Last accessed July, 2009.
[33] Klaming, L., Van Veenen, J., Leenes, R. (2008). I want the opposite of what you want: summary of a study on the reduction of fixed-pie perceptions in online negotiations. In Expanding the horizons of ODR, Proceedings of the 5th International Workshop on Online Dispute Resolution, Firenze, Italy, pp. 84-94.
[34] Goodman, J.W. (2003). The pros and cons of online dispute resolution: an assessment of cyber-mediation websites, in Duke Law and Technology Review.
[35] De Vries, BR., Leenes, R., Zeleznikow, J. (2005). Fundamentals of providing negotiation support online: the need for developping BATNAs. Proceedings of the Second International ODR Workshop, Tilburg, Wolf Legal Publishers, 59-67.
[36] Peruginelli, G., Chiti, G. (2002) Artificial Intelligence in alternative dispute resolution. Proceedings of the Workshop on the law of electronic agents – LEA.

[37] Katsch, E. & Rifkin, J. (2001). Online dispute resolution – resolving conflicts in cyberspace. Jossey-Bass Wiley Company, San Francisco.
[38] Bellifemine, F., Poggi, A., Rimassa, G. (2008). Developing Multi-agent Systems with JADE, Springer.
[39] Foundation for Intelligent Physical Agents: FIPA ACL Message Structure Specification (2002). Available at http://www.fipa.org/specs/fipa00061, accessed in July 2011.
[40] Riesbeck, C., and Bain, W. (1987). A Methodology for Implementing Case-Based Reasoning Systems, Lockheed.
[41] Slade, S. (1991). Case-Based Reasoning: A Research Paradigm. AI Magazine. Vol. 12.
[42] Alterman, R. (1989). Panel discussion on case representation. In Proceedings of the Second Workshop on Case-Based Reasoning, Pensacola Beach. FL, US.
[43] Watson, I., Marir, F. (1994). Case-based reasoning: A Review. In: Knowledge Engineering Review, vol. 9, pp. 327–354.
[44] Carneiro D., Novais P., Andrade F., Zeleznikow J., Neves J. (2009). The Legal Precedent in Online Dispute Resolution. In Legal Knowledge and Information Systems, ed. Guido Governatori, Proceedings of the Jurix 2009 - the 22nd International Conference on Legal Knowledge and Information Systems, IOS press, ISBN 978-1-60750-082-7, pp. 47—52.
[45] Fisher, R. Ury, W. (1981). Getting to YES: Negotiating Agreement Without Giving In, Boston: Haughton Mifflin.
[46] Thomas, K., Kilmann, R. (1974) Conflict and Conflict Management. Available at http://www.kilmann.com/conflict.html, accessed in July 2011.
[47] Andrade, F., Barbieri, D., Carneiro, D., Novais, P. (2010) Artificial Intelligence Applications in ODR: The UMCourt Project, Proceedings of the 17th Annual Northwest Dispute Resolution Conference, University of Washington, USA.
[48] Thiessen, E.M. (1993). ICANS: An Interactive Computer-Assisted Multi-party Negotiation Support System. PhD Dissertation, School of Civil & Environmental Engineering, Cornell University, Ithaca, NY.
[49] Follett, M. P. (1940). Constructive Conflict. In Metcalf , H. C. and Urwick, I. L. (eds.). Dynamic Administration: The collected papers of Mary Parker Follett, pp. 30-49, New York: Harper.
[50] Waterman, D.A., Peterson, M. (1980). Rule-based models of legal expertise. In Proceedings of the First National Conference on AI, Stanford University.
[51] Cáceres, E. (2008). EXPERTIUS: A Mexican Judicial Decision-Support System in the Field of Family law. In Francesconi, E. B. E., Sartor, G., & Tiscornia, D. (Eds.), Legal Knowledge and Information Systems, pp. 78-87. IOS Press.
[52] Carneiro D., Novais P., Costa R., Neves J. (2010). Enhancing the Role of Multi-agent Systems in the Development of Intelligent Environments. In Advances in Intelligent and Soft Computing, Vol. 71, Springer-Verlag, ISBN 978-3-642-12432-7, pp. 123-130.
[53] Steinbach, M., Tan P. N. and Kumar, V. Eds. (2005). Introduction to Data Mining. Pearson Addison Wesley.
[54] Tang, M., Zhou, Y., Li, J., Wang, W., Cui, P., Hou, Y., Luo, Z., Li, J., Lei, F., Yan, B. (2010). Exploring the wild birds' migration data for the disease spread study of H5N1: a clustering and association approach. In Knowledge and Information Systems, Springer London, pp. 1-25, DOI: 10.1007/s10115-010-0308-x.
[55] Hasan, M., Salem, S., Zaki, M. (2010). SimClus: an effective algorithm for clustering with a lower bound on similarity. In Knowledge and Information Systems, Springer London, pp. 1-21, DOI: 10.1007/s10115-010-0360-6.
[56] Lynda, T., Mohand, B., Mariam D. (2010). Evaluation of contextual information retrieval effectiveness: overview of issues and research. In Knowledge and Information Systems, Vol. 24, Number 1, 1-34, Springer London, DOI: 10.1007/s10115-009-0231-1.
[57] Wu, Xindong et al. (2007). Top 10 algorithms in data mining. In In Knowledge and Information Systems, Vol. 14, Issue 1, Springer-Verlag New York, 10.1007/s10115-007-0114-2.