DISARM: A Social Distributed Agent Reputation Model based on Defeasible Logic

Kalliopi Kravari, Nick Bassiliades*

Dep. of Informatics, Aristotle University of Thessaloniki, GR-54124, Thessaloniki, Greece
{kkravari, nbassili} AT csd.auth.gr

ABSTRACT Agents act in open and thus risky environments with limited or no human intervention. Making the appropriate decision about who to trust in order to interact with is not only necessary but it is also a challenging process. To this end, trust and reputation models, based on interaction trust or witness reputation, have been proposed. Yet, they are often faced with skepticism since they usually presuppose the use of a centralized authority, the trustworthiness and robustness of which may be questioned. Distributed models, on the other hand, are more complex but they are more suitable for personalized estimations based on each agent's interests and preferences. Furthermore, distributed approaches allow the study of a really challenging aspect of multi-agent systems, that of social relations among agents. To this end, this article proposes DISARM, a novel distributed reputation model. DISARM treats Multi-agent Systems as social networks, enabling agents to establish and maintain relationships, limiting the disadvantages of the common distributed approaches. Additionally, it is based on defeasible logic, modeling the way intelligent agents, like humans, draw reasonable conclusions from incomplete and possibly conflicting (thus inconclusive) information. Finally, we provide an evaluation that illustrates the usability of the proposed model.

Keywords: Multi-agent Systems; Agent Reputation; Distributed Trust Management; Logic-Based Approach; Defeasible Reasoning, Semantic Web

1 Introduction

Intelligent Agents (IAs) act in open and thus risky environments, hence making the appropriate decision about the degree of trust that can be invested in a certain partner is vital yet really challenging [60]. Over the last few years, scientific research in this field has significantly increased. Most researchers tend to consider trust and reputation as key elements in the design and implementation of modern multi-agent systems (MASs). However, there is still no single, accepted definition of trust within the research community, although it is generally defined as the expectation of competence and willingness to perform a given task. Broadly speaking, trust has been defined in a number of ways in the literature, depending on the domain of use. Among these definitions, there is one that can be used as a reference point for understanding trust, provided by Dasgupta [24]. According to Dasgupta, trust is a belief an agent has that the other party will do what it says

*Corresponding author. Nick Bassiliades, Department of Informatics, Aristotle University of Thessaloniki, GR-54124 Thessaloniki, Greece. E-mail: nbassili@csd.auth.gr, Tel: +302310997913, Fax: +302310998419.

it will (being honest and reliable) or reciprocate (being reciprocative for the common good of both), given an opportunity to defect to get higher payoffs.

Trust, however, is much more than that; the uncertainties found in the modern MASs present a number of new challenges. More specifically, MASs are open used to model distributed systems, sometimes large-scaled, which means that the agents represent different stakeholders that are likely to be self-interested and might not always complete tasks requested from them. Moreover, in an open system, usually no central authority can control all the agents, which means that agents can join and leave at any time. The problem is that this allows agents to change their identity and re-enter, avoiding punishment for any past mistakes. One, more, risky feature of open systems is that when an agent first enters the system has no information about the other agents in that environment. Given this, the agent is likely to be faced with a large amount of possible partners with a different degree of efficiency and/or effectiveness.

Systems that present these challenges can be found in a variety of domains. For example, as ambient intelligence is becoming more widespread, sensors and smart devices are included more and more often in new buildings and vehicles. Technologies and applications for green growth, such as smart buildings and grids, attracted the attention of researchers and industry over the last years. In this context, applications in a number of fields have been already developed and used. Transportation and logistics, environmental monitoring and energy control systems are just some of these fields. Actually, intelligent agents are increasingly used in networking and mobile technologies in order to achieve, among others, automatic and dynamic behavior, high scalability and self-healing networking. Multi-agent systems are also applied in the real world in many other cases; films and computer games use agent technology. Other applications include health care and medical diagnostics, crisis management and coordination of defense systems. Especially nowadays that terrorism has become a considerable issue, networks of sensors, smart devices or UAVs (unmanned aerial vehicles) equipped with intelligent agents would be able to communicate exchanging data and perform inferences about terrorist attacks or other emergencies. In other words, such sensors and devices, being able to communicate with each other, form a multi-agent system, which is subject to the principles and challenges of agent technology. [18, 50, 73, 76]

Ubiquity, decentralization, openness, dynamism and uncertainty are the main challenges encountered in most of the above systems and applications. These challenges lead to two important but yet difficult to handle issues; decision making in uncertain and partially observable environment and agent cooperation in such environments. The underlying reason for this is the fact that intelligent agents are “created” by self-interested individuals, companies, organizations or governments. To this end, a number of researchers were motivated by the understanding that some individuals, and thus their agents, may be dishonest, focusing eventually their efforts on agents’ reputation. In general, reputation is the opinion of the public towards an agent. Reputation allows agents to build trust, or the degree to which one agent has confidence in another agent, helping them to establish relationships that achieve mutual benefits. Hence, reputation (trust) models help agents to decide who to trust, encouraging trustworthy behavior and deterring dishonest participation by providing the means through which reputation and ultimately trust can be quantified [51, 62]. Hence, as intelligent agents are gradually enriched with Semantic Web technology [71, 34, 10, 9], acting on behalf of their users with limited or no human intervention, their ability to perform assigned tasks is scrutinized. To this end, plenty of trust and reputation models have been proposed in different perspectives, yet they often presuppose the use of a centralized authority [51, 60]. Although such reputation mechanisms are more popular, they are

usually faced with skepticism, since in open MASs agents represent different owners, who may question the trustworthiness and the robustness of a central authority.

On the other hand, distributed reputation models are typically more complex and require a lot of communication in order agents to exchange their ratings. These models have no centralized system manager; hence each agent has to overcome the difficulty of locating ratings and develop somehow a subjective estimation by itself using its own resources. No global or public reputation exists. The reputation built in this way is thus personalized and sometimes difficult to determine. However, a distributed reputation system is more flexible in building agents' reputation, since it is quite easy for an agent to develop differentiated trust in other agents based on its interests and purposes. Yet, beyond the traditional choice of centralized or distributed approach, there is an even more challenging decision; what should be taken into account in order to estimate the reputation of an agent, interaction trust or witness reputation [5, 51]. In other words, an agent's direct experience or reports provided by others, respectively.

Broadly speaking, both approaches have limitations. For instance, if the reputation estimation is based only on direct experience, it would require a long time for an agent to reach a satisfying estimation level. This is because, when an agent enters an environment for the first time, it has no history of interactions with the other agents in the environment. Thus, it needs a long time to reach a sufficient amount of interactions that could lead to sufficient information. On the other hand, models based only on witness reports could not guarantee reliable estimation as self-interested agents could be unwilling or unable to sacrifice their resources in order to provide reports. Hence, models based only on one or the other approach typically cannot guarantee stable and reliable estimations. To this end, in order to overcome these drawbacks, a number of hybrid models that combine both interaction trust and witness reputation were proposed [5, 36, 37, 51]. However, most of hybrid models either have a fixed proportion of their participation in the final estimation or leave the choice to the final user. Although these approaches have significant advantages, sometimes they may lead to misleading estimations. Users may have little or no experience and thus take wrong decisions that could lead to wrong assessments, whereas fixed values provide just generic estimations. Our goal is not to estrange the users from the decision making process, but to help them, and their agents, to make better decisions.

To this end, this article proposes a novel distributed reputation model, called DISARM, that combines both interaction trust and witness reputation. DISARM is a knowledge-based approach, based on well-established estimation parameters [19, 20, 32, 35, 38, 69, 67], that provides a more intuitive method for non-technical users. More specifically, its aim is to reduce the disadvantages of the common distributed hybrid approaches, such as the difficulty in locating ratings, and provide a mechanism for modeling the way intelligent agents, like humans, draw reasonable conclusions from incomplete and possibly conflicting (thus inconclusive) information. This is achieved by designing and implementing a reputation mechanism based on social principles and defeasible logic. Concerning the social aspect of the model, DISARM proposes an approach where agents are enabled to establish, through their interactions, and maintain relationships, much as individuals do in real life. More specifically, DISARM considers agents acting in the environment as a social network which determines the proximity relationships among them. In this context, all known agents create a network, which is expanded whenever new agent interactions take place in the environment. The advantage of this approach is that it allows agents to communicate with previously known and well-rated agents, locating, quite fast, ratings with small bandwidth cost.

Concerning the modeling mechanism, although it is logic-independent, DISARM proposes the use of defeasible logic, a logic that has the notion of rules that can be defeated, allowing an

existing belief to turn false, making it nonmonotonic [53, 58]. In a fundamental sense, nonmonotonic logics occupy undoubtedly prominent position among the disciplines investigating intelligent reasoning about complex and dynamic situations. Thus, permitting agents to arrive at defeasible conclusions, leads to more realistic assessments similar to human reasoning. Additionally, defeasible logic is part of a more general area of research, defeasible reasoning, which is notable for its low computational complexity [49].

As a result, DISARM, combining a set of features, is able to address many of the challenges of trust management. It has three main features: it is distributed, nonmonotonic and social. It avoids single point of failure since it is a distributed approach and, thus, it does not rely on an individual entity. Furthermore, being nonmonotonic, it allows invalidating previous conclusions in the light of new information while by using features from social networks, it enables more efficient computation compared to other distributed approaches. DISARM combines in a new promising way existing formal methods and reasoning machineries, such as defeasible logic, in order to model reputation and trust aspects while it proposes novel metrics that are able to support accurate reputation assessment.

Moreover, we provide an evaluation that illustrates the usability of the proposed model. The rest of the article is organized as follows. In Section 2, we present a brief overview of defeasible logic. Section 3 presents DISARM and its contribution. In Section 4, DISARM's evaluation is presented, demonstrating the added value of the approach. Section 5 discusses related work, and Section 6 concludes with final remarks and directions for future work.

2 Defeasible logic

Defeasible logic (DL), introduced by Nute [54, 55] with a particular concern about efficiency and implementation, is part of a more general area of research, namely defeasible reasoning [58, 57]. Over the years the logic has been developed and extended while several variants have been proposed. Yet, DL remains a simple and efficient rule based nonmonotonic formalism that deals with incomplete and conflicting information. More specifically, DL has the notion of rules that can be defeated; hence it derives plausible conclusions from partial and sometimes conflicting information. These conclusions, despite being supported by the currently available information, could nonetheless be rejected in the light of new, or more refined, information.

Compared to other more mainstream nonmonotonic approaches, e.g. [61, 27], this approach offers among others enhanced representational capabilities and low computational complexity. Moreover, DL in contrast with traditional deductive logic, allows the addition of further propositions to make an existing belief false, making it nonmonotonic [43]. In a fundamental sense, nonmonotonic logics occupy undoubtedly prominent position among the disciplines investigating intelligent reasoning about complex and dynamic situations / environments. Hence, one of the main interests in DL is in the area of agents [28]. DL, being a nonmonotonic logic, is capable of modeling the way intelligent agents, like humans, draw reasonable conclusions from inconclusive information, leading to more realistic conclusions and assessments similar to human reasoning.

Hence, DL can be and it is adapted in a variety of applications involving decision making or negotiation. For instance, DL was successfully applied in a robotic domain for knowledge representation and reasoning about which task to perform next [25]. Negotiation is another domain that benefits from DL, since it is flexible, has efficient implementations and provides a formal basis for analysis. As a result, DL has been adapted in many e-Commerce cases that include negotiation,

such as brokering and bargaining [29, 68]. DL can be used even in order to provide a formalism for specifying authorization policies of a dynamic system, such as in [66].

Knowledge in DL is represented in terms of facts, rules and superiority relations. Facts are indisputable statements, represented either in form of states of affairs (literal and modal literal) or actions that have been performed. A rule describes the relationship between a set of literals (premises) and a literal (conclusion). Rules are divided into strict rules, defeasible rules and defeaters. Strict rules are rules in the classical sense, i.e. whenever the premises are indisputable, e.g. facts, then so is the conclusion. Thus, they can be used for definitional clauses. Defeasible rules, on the other hand, are rules that can be defeated by contrary evidence. Defeaters are rules that cannot be used to draw any conclusions. Their only use is to prevent some conclusions. Finally, the superiority relation is a binary relation defined over the set of rules, which determines the relative strength of two (conflicting) rules, i.e. rules that infer conflicting literals.

The form of rules, in symbolic and d-POSL syntax [42], are presented in Table 1. d-POSL syntax, which is used throughout this article, is a grammar that maintains all the critical components of the Positional-Slotted Language (POSL), extended with components necessary for defeasible logic. More specifically, POSL and, as a result, d-POSL are ASCII languages that integrate Prolog's positional and Frame-logic's slotted syntaxes for representing knowledge (facts and rules) in the Semantic Web [15]. d-POSL syntax allows us to represent defeasible rule bases in a compact and human-readable way, since it is faster to write and easier to read. Variables in d-POSL are denoted with a preceding "?" while rule types ("strict", "defeasible", "defeater") are expressed via binary infix functors (":-", ":=", ":~"). The correspondence between d-POSL syntax and the traditional defeasible rule syntax is presented in Table 1.

Table 1 Rules in Defeasible Logic.

RULE TYPE	***RULE FORM***	***D-POSL SYNTAX***
STRICT RULES	A1, … , An → B	B:-A1, … , An
DEFEASIBLE RULES	A1, … , An => B	B:=A1, … , An
DEFEATERS	A1, … , An ~> B	B:~A1, … , An

Non-monotonic reasoning systems represent and reason with incomplete information where the degree of incompleteness is not quantified. A very simple and natural way to represent such incomplete information is with a defeasible rule of the form "antecedent ⇒ consequent"; with the meaning that provided there is no evidence against the consequent, the antecedent is sufficient evidence for concluding the consequent [13].

Being nonmonotonic, defeasible logics deal with potential conflicts (inconsistencies) among knowledge items. Thus they contain classical negation, contrary to usual logic programming systems. They can also deal with negation as failure (NAF), the other type of negation typical of nonmonotonic logic programming systems; in fact, [72] argues that the Semantic Web requires both types of negation. NAF is not included in the object language of DL. However, as [3, 4] show, it can be easily simulated when necessary. Thus, we may use NAF in the object language and transform the original knowledge to logical rules without NAF exhibiting the same behavior [72]. Furthermore, note that negated literals can appear both in the head and in the body of a rule.

The main concept in DL is that it does not support contradictory conclusions, but it tries to resolve conflicts. Hence, in cases where there is some support for concluding A, but there is also

support for concluding ¬A (the negation of A), no conclusion is derived unless one of the two rules that support these conflicting conclusions has priority over the other. This priority is expressed through a superiority relation among rules which defines priorities among them, namely where one rule may override the conclusion of another rule. Yet, conclusions can be classified as definite or defeasible. A definite conclusion is a conclusion that cannot be withdrawn when new information is available. A defeasible conclusion, on the other hand, is a tentative conclusion that might be withdrawn in the future. In addition, the logic is able to tell whether a conclusion is or is not provable, hence there are four possible types of conclusions; positive definite, negative definite, positive defeasible and negative defeasible.

A formal way of computing conclusions in DL can be found in [3]. More specifically, a conclusion in a D (defeasible) theory is a tagged literal and may have, as described above, one of the following forms:

$+\Delta q$, meaning that q is definitely provable in D.

$+\partial q$, meaning that q is defeasibly provable in D.

$-\Delta q$, meaning that q has proved to be not definitely provable in D.

$-\partial q$, meaning that q has proved to be not defeasibly provable in D.

In order to prove $+\Delta q$ (positive definite conclusion), a proof for q consisting of facts and strict rules needs to be established. Whenever a literal is definitely provable, it is also defeasibly provable. In that case, the defeasible proof coincides with the definite proof for q. Otherwise, in order to prove $+\partial q$ (positive defeasible conclusion) in D, an applicable strict or defeasible rule supporting q must exist. In addition, it should also be ensured that the specified proof is not overridden by contradicting evidence. Therefore, it has to be guaranteed that the negation of q is not definitely provable in D. Successively, every rule that is not known to be inapplicable and has head ~q has to be considered. For each such rule s, it is required that there is a counterattacking rule t with head q that is applicable at this point and s is inferior to t.

In order to prove $-\Delta q$ (negative definite conclusion) in D, q must not be a fact and every strict rule supporting q must be known to be inapplicable. If $-\Delta q$ is proved, then $-\partial q$ also holds. Otherwise, in order to prove that $-\partial q$ (negative defeasible conclusion), it must firstly be ensured that $-\Delta q$. Additionally, one of the following conditions must hold: (i) None of the rules with head q can be applied, (ii) It is proved that $-\Delta \sim q$, and (iii) There is an applicable rule r with head ~q, such that no possibly applicable rule s with head q is superior to r.

In this context, a special case of conflict is between different positive literals, all derived by different defeasible rules, whereas only one should be derived. “Conflicting literals” are defined through a conflict set and the conflict is resolved through superiorities. Conflicting literals are actually an important type of conflicting evidence in defeasible reasoning. This type of conflict arises when two or more opposite conclusions (positive literals) can be derived. Yet, just one of them can be considered true since only a piece of information or state can be valid at a specific time point. For instance, consider the case of academic grading. Generally, there are four discrete types of grades, {Excellent, Very Good, Good, Withdrawal}, and each student receives only one of them. Hence, these grades form a set of conflicting literals [3 , 14, 12].

3 DISARM

The proposed model is called DISARM and it is a distributed, hybrid, rule-based reputation model. DISARM uses defeasible logic in order to combine in a practical way all available ratings, both those based on the agent's personal experience and those provided by known and/or unknown third parties. This model aims not only at reducing the disadvantages of the common distributed approaches, such as the difficulty in locating ratings, but mainly it aims at improving the performance of the hybrid approach by providing an intuitive decision making mechanism. DISARM aims at providing a distributed mechanism based on defeasible logic that would be able to model the way humans think, infer and decide.

3.1 Main principles of the DISARM Model

For purposes of better understanding, we present here the main principles of our model. First of all, DISARM has no centralized authority since it is a distributed model. Hence, it is each agent's responsibility to locate ratings and use the model. In this context, even if more than one agents possess the same ratings, they will probably come out with different estimations and as a result they will take different decisions.

Time, evolution over time in particular, is an important issue since it reflects the behavior of an agent. More specifically, in dynamic environments such as MASs, agents may change their objectives at any time. For instance, a typical dishonest agent could provide quality services over a period to gain a high reputation score, and then, profiting from that high score could provide low quality services. Hence, time should be and it is taken into account in the proposed model. Yet, DISARM allows agents to decide on their own about what they consider important. To this end, it is up to each agent's strategy to determine the impact of time in their decisions. Agents could take into account all the available ratings or only the latest; e.g. those referred to last week, last month or last year.

Taking into account the latest ratings leads undoubtedly to an up-to-date overview, however it could be misleading. For instance, in this limited time period, a typical dishonest agent could temporary improve its behavior or, on the other hand, a typical reliable agent, facing a problem, could temporary act faulty, transformed into a mercenary and malicious agent. Hence, it is a risk to take into account only part of the available ratings, although there is sometimes significant gain in time and computational cost. In contrary, taking into account all available ratings leads to an overview of an agent's behavior history but it costs in terms of storage space, execution time and computational power.

DISARM, however, is a distributed model which means that locating ratings is a quite challenging process. The rating records could always be there but usually they are unreachable since various agents may join or leave the system at any time. For instance, sometimes only a few ratings are available; e.g. personal experience could be missing and/or appropriate witnesses could be difficult to locate. On the other hand, sometimes there is a large amount of available ratings but taking all of them into account has significant computational cost. Moreover, these ratings may significantly differ. In this context, DISARM integrates an indication of how likely is the assessment to be proved correct based on the variability of ratings that were taken into account. In other words, DISARM allows agents to be informed about the possibility of wrong estimation and loss.

Another important issue that DISARM deals with is the trust relationships that agents build and maintain over time, much as individuals do in real world. For instance, if an agent is satisfied with a partner, probably it will prefer to interact again with that partner in the future. On the other hand, if it is disappointed by a partner, it will avoid interacting again with that partner. To this end, DISARM proposes the use of two lists, called whitelist and blacklist. Each agent stores in its whitelist the names of its favored partners while in its blacklist it stores those that should be avoided. The decision about who will be added in each list is taken by the agent itself. More specifically, each agent is equipped with a rule-based decision-making logic which enables it to decide upon its partners, adding them, if necessary, to the appropriate list. Hence, it will be quite easy for the agent to locate a well-known old partner that will do the job and at the same time avoid a fraud. Moreover, a user is much more likely to believe statements from a trusted acquaintance than from a previously known dishonest agent or a stranger.

Finally, in addition to the difficulty to locate ratings there is also a difficulty to locate really useful ratings. For instance, sometimes agents are involved in important and crucial for them interactions whereas sometimes they are involved in simple interactions of minor importance. Hence, the question is which of them should be taken into account in order to get a representative estimation. To this end, DISARM adopts the use of two more parameters for each rating; namely importance and confidence. Importance indicates how critical the transaction was for the rating agent while confidence gives an estimation of the agent's certainty for that rating.

3.2 Rating parameters

Taking into account the proper parameters for an assessment is a really challenging task. They should be carefully chosen in order to reflect the agents' abilities. Besides, an efficient decision making mechanism has to rely on carefully selected data and a straightforward and efficient rating procedure. Although, a thorough overview of related literature is out of the scope of this article, we tried to catch out parameters, or factors for others, that are usually referred either explicitly or implicitly in reputation models and metrics, e.g. [19, 20, 32, 35, 38, 69, 67]. To this end, DISARM uses for its needs six properties; namely response time, validity, completeness, correctness, cooperation and outcome feeling.

Response time refers to the time that an agent needs in order to complete the tasks that it is responsible for. Time is the only parameter that is always taken into account in the literature. *Validity* describes the degree that an agent is sincere and credible. An agent is sincere when it believes what it says, whereas it is credible when what it believes is true in the world. Hence, an agent is valid if it is both. Validity is not always such called, yet in most cases there are parameters that attempt to indicate how sincere and/or credible an agent is. *Completeness*, on the other hand, describes the degree that an agent says what it believes while what it believes is true in the world. In other words, completeness is the inverse of validity, indicating how honest and realistic an agent is. Completeness is usually implicitly referred, as an attempt to rate dishonest and fraud behavior.

Moreover, *correctness* refers to an agent's providing services or tasks. An agent is correct if its provided service or task is correct with respect to a specification. Correctness, no matter how it is called, is, actually, the second most used parameter after time. *Cooperation* is the willingness of an agent who is being helpful by doing what is wanted or asked for. Cooperation is not, usually, handled as separate parameter, however, it is an important feature in distributed social environments, such as MASs. Finally, the *outcome feeling* is a general feeling of satisfaction or dissatisfaction related to the transaction outcome; namely it indicates if the transaction was easy

and pleasant with a satisfying result or not. Usually, it is referred as the degree of request fulfillment.

However, although these six parameters are, usually, taken into account in one way or another, they are not necessarily binding. Some of them could be replaced by other more domain-specific parameters depending on the domain of use, e.g. agents acting in E-Commerce transactions may include parameters about transaction security, payment variety (accepted ways of payment) and existence of digital certificates for electronic authentication. Yet, our intention, here, in the context of DISARM, is to provide general purpose parameters that will be able to reflect the common critical characteristics of each agent in the community. In other words, DISARM takes into account parameters that can provide an overview of each agent's behavior. Hence, consider an agent A establishing an interaction with an agent X; agent A can evaluate the other agent's performance and thus affect its reputation. The evaluating agent (A) is called *truster* whereas the evaluated agent (X) is called *trustee*. Of course, for some interactions an agent can be both truster and trustee, since it can evaluate its partner while it is evaluated by that partner at the same time. After each interaction in the environment, the truster has to evaluate the abilities of the trustee in terms of response time, validity, completeness, correctness, cooperation and outcome feeling. DISARM, however, is a distributed model hence truster does not have to report its ratings but just to save them for future use.

Yet, in order to remember how important was the transaction for it and how confident it was for its rating, the agent has to associate two more values to the rating, as was already discussed, namely confidence and importance. Additionally, since time is considered an important aspect in DISARM's decision making process, each rating is associated with a time stamp (t), e.g. in the form YYMMDDHHMMSS or YYMMDD or HHMMSS or it can be represented even as an integer in case of experimental simulations, indicating the transaction's time point. Hence, taking the above into account the truster's rating value (r) in DISARM is a tuple with eleven elements: *(truster, trustee, t, response time, validity, completeness, correctness, cooperation, outcome feeling, confidence, transaction value)*. Notice, that although each truster agent stores its own ratings, we include the variable truster in the rating value. This is because the truster may forward its ratings to other agents; hence, these agents should be able to identify the rating agent for each rating they receive. In DISARM, the rating values vary from 0.1 (terrible) to 10 (perfect); $r \in [0.1, 10]$, except confidence and transaction values that vary from 0 (0%) to 1 (100%).

3.3 Rule-based decision mechanism

Defining the rating values is the first step towards an efficient reputation model, the core of the approach, however, is its decision making mechanism. The distributed reputation models have invariably to deal with a range of complex issues related to the decision making process, such as locating ratings. Hence, DISARM aims at providing a trust estimation procedure much as individuals do in real world, where they build and maintain trust relationships over time. To this end, DISARM simulates their decision making process, proposing a set of strict and defeasible rules, in a practical, intuitive approach.

3.3.1 Rating procedure

First of all, shortly after an interaction ends each agent evaluates its partner in terms of response time, validity, completeness, correctness, cooperation and outcome feeling. Then it adds its confidence and a value indicating the importance of the transaction (transaction value). When all values are got together, the rating agent (truster) adds its name, the trustee's name and the current

time point (t), forming the final rating value (r) as a tuple with eleven elements. This tuple is presented below in the compact d-POSL syntax [42] of defeasible RuleML [7]. A syntax that will be used throughout this article in order to express in a compact way the data (ratings) and rules (strict and defeasible rules used in the decision making process) of our approach. To this end, the truster's rating (r) is the fact:

rating(*id*→id_x, *truster*→*?a*, *trustee*→*?x*, *t*→*?t*,
response_time→$?resp_x$, *validity*→$?val_x$, *completeness*→$?com_x$,
correctness→$?cor_x$, *cooperation*→$?coop_x$, *outcome_feeling*→$?outf_x$,
confidence→$?conf_x$, *transaction_value*→$?trans_x$).

Additionally, an example rating provided by agent (A) truster for the agent (X) trustee could be:

rating(*id*→*1*, *truster*→*A*, *trustee*→*X*, *t*→*140630105632*, *response_time*→*9*, *validity*→*7*,
completeness→*6*, *correctness*→*6*, *cooperation*→*8*,
outcome_feeling→*7*, *confidence*→*0.9*, *transaction_value*→*0.8*).

Next, truster stores this rating to its repository. However, as already mentioned, agents compliant with DISARM use two lists, additionally to their rating repository; whitelist and blacklist. More specifically, these lists are two separate repositories, one for storing promising partners (whitelist) and one for those partners that should be avoided (blacklist). Hence, truster has also to decide whether it should add the trustee to its white (or black) list or not. Obviously, the decision is based on what it is considered as a promising (or terrible on the other hand) partner. A partner is promising if it acts responsibly and it provides high quality services or products. A partner is responsible if it is cooperative, responds fast and leaves a positive feeling at the end of the transaction.

Of course, each agent has a different degree of tolerance, thus, what may be fast for an agent could be slow for another. Hence, each agent has some thresholds that determine the lowest accepted value for each parameter; namely response time, validity, completeness, correctness, cooperation and outcome feeling. As a result, the behavior of an agent is characterized as good if parameter values are higher than thresholds or bad if they are not. In this context, rule r_1, presented below, indicates that if all values are higher than the truster's associated thresholds then the trustee's behavior is considered good. Yet, this rule is strict and quite rare, since humans, and thus their agents, usually are not so rigorous.

r_1: *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *all*) :-
response_time_threshold(*?resp*), *validity_threshold*(*?val*),
completeness_threshold(*?com*), *correctness_threshold*(*?cor*),
cooperation_threshold(*?coop*), *outcome_feeling_threshold*(*?outf*),
rating(*id*→$?id_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*,
response_time→$?resp_x$, *validity*→$?val_x$, *completeness*→$?com_x$,
correctness→$?cor_x$, *cooperation*→$?coop_x$, *outcome_feeling*→$?outf_x$),
$?resp_x$>*?resp*, $?val_x$>*?val*, $?com_x$>*?com*, $?cor_x$>*?cor*, $?coop_x$>*?coop*, $?outf_x$>*?outf*.

The above rule (r_1) consists of a number of clauses, namely six user-defined thresholds (one per parameter, e.g threshold for the response time: *response_time_threshold*(*?resp*)) and a rating that includes information about time (when that rating was reported), the involved agents (truster

and trustee) and the parameter values (truster's opinion about trustee's performance related to six parameters). Given these clauses, r_1 compares each parameter value with the associated user-defined threshold (e.g. $?resp_x > ?resp$, where $?resp_x$ is the rating value for the response time parameter and *?resp* is the threshold). If these (six) comparisons, one per parameter, reveal that the rating values are greater than thresholds then r_1 concludes that at a specific time point (*?t*), truster A (*?a*) considers trustee's (*?x*) behavior good for all parameters (reasons).

Yet, usually, truster classifies its partner in relation to a reason, e.g. response time. In this context, rules r_2 to r_7 present the group of rules that characterize the behavior of an agent as good depending on a specific reason.

r_2: *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *response_time*) :-
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *response_time*→*?resp*$_x$),
response_time_threshold(*?resp*), *?resp*$_x$>*?resp*.

r_3: *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *validity*):-
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *validity*→*?val*$_x$),
validity_threshold(*?val*), *?val*$_x$>*?val*.

r_4: *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason*→ *completeness*) :-
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *completeness*→*?com*$_x$),
completeness_threshold(*?com*), *?com*$_x$>*?com*.

r_5: *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason*→ *correctness*):-
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *correctness*→*? cor*$_x$),
correctness_threshold(*?cor*), *?cor*$_x$>*?cor*.

r_6: *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason*→ *cooperation*) :-
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *cooperation*→*?coop*$_x$),
cooperation_threshold(*?coop*), *?coop*$_x$>*?coop*.

r_7: *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *outcome_feeling*):-
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *outcome_feeling*→*?outf*$_x$),
outcome_feeling_threshold(*?outf*), *?outf*$_x$>*?outf*.

For instance, r_7 consists of two main clauses: a rating that includes the time that the rating was reported the rating, the involved agents (truster and trustee) and a value related to the outcome feeling parameter (since this is the only parameter that is taken into consideration), as well as a user-defined threshold for the outcome feeling. Given these clauses, r_7 compares the parameter value with the threshold ($?outf_x > ?outf$, where $?outf_x$ is the rating value and *?outf* is the threshold). If the rating value is greater than the threshold then r_7 rule concludes that at a specific time point (*?t*), truster A (*?a*) considers trustee's (*?x*) behavior good as far as it concerns the outcome feeling of a transaction.

However, from each agent's perspective a parameter could be more important than others. For instance, an agent could consider response time the most important aspect, perhaps not the only one, in deciding whether its partner could be characterized good or bad. In this context, rules r_2 to r_7 could be "replaced" by defeasible rules (r_2' - r_7'), while a priority relationship among all or some of them will define which reason is eventually more important. Next, such an example is presented, where a truster considers response time and validity important reasons while validity is considered

more important than response time. Of course, any potential theory could be formed, namely rule combination and priority relationship regarding the reasons, in order to represent truster's personal preferences (private strategy).

r_2': *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *response_time*) :=
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *response_time*→*?resp*$_x$),
response_time_threshold(*?resp*), *?resp*$_x$>*?resp*.

r_3': *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *validity*):=
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *validity*→*?val*$_x$),
validity_threshold(*?val*), *?val*$_x$>*?val*.

$r_2' > r_3'$

The conflict set for the above theory (rules r_2', r_3') is formally determined as follows, meaning that only one reason for good behavior, the most important one, is kept at any time:

C[*good_behavior(time* → *?t, truster*→ *?a, trustee*→ *?x, reason* → *?reason)*] =
{ ¬*good_behavior(time* → *?t, truster*→ *?a, trustee*→ *?x, reason* → *?reason)*} ∪
{ good_behavior(time → ?t, truster→ ?a, trustee→ ?x, reason → ?reason1)
| *?reason1* ≠ *?reason* }

Of course, truster could classify its partner in relation to more than one reasons. For instance, rule r_8 describes such a case. According to that "an agent has a good behavior if it has good rating in at least three parameters".

r_8: *good_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *at_least_3*):-
good_for(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *?r1*),
good_for(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *?r2*),
good_for(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *?r3*),
?r1 ≠ *?r2*, *?r2* ≠ *?r3*, *?r3* ≠ *?r1*.

On the other hand, trustee's behavior is considered disappointing and, thus, bad in relation to a reason/parameter, if trustee's rate for this parameter is lower than the truster's thresholds. A lenient agent would expect all values to be lower than its thresholds in order to characterize the behavior of an agent as bad. In this context, r_9 presents such a case.

r_9: *bad_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *all*) :-
response_time_threshold(*?resp*), *validity_threshold*(*?val*),
completeness_threshold(*?com*), *correctness_threshold*(*?cor*),
cooperation_threshold(*?coop*), *outcome_feeling_threshold*(*?outf*),
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*,
response_time→*?resp*$_x$, *validity*→*?val*$_x$, *completeness*→*?com*$_x$,
correctness→*?cor*$_x$, *cooperation*→*?coop*$_x$, *outcome_feeling*→*?outf*$_x$),
?resp$_x$<=*?resp*, *?val*$_x$<=*?val*, *?com*$_x$<=*?com*, *?cor*$_x$<=*?cor*,
?coop$_x$<=*?coop*, *?outf*$_x$<=*?outf*.

On the other hand, it may be the case that a truster considers the behavior of an agent as bad based on one specific reason alone. For instance, r_{10} presents such a case, based on response time alone. Of course, *response_time* (reason) could be replaced by anyone of the rest of the parameters, namely *validity*, *completeness*, *correctness*, *cooperation* and *outcome_feeling*. Furthermore, an agent could consider important more than one reasons (e.g. validity and correctness, combined). Then, similarly as rules related to good behavior, a theory, namely a set of rules (one per reason) and a priority relationship among these rules, will represent the preferences of the agent.

r_{10}: *bad_behavior*(*time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *reason* → *response_time*) :-
rating(*id*→*?id*$_x$, *time* → *?t*, *truster*→ *?a*, *trustee*→ *?x*, *response_time*→*?resp*$_x$),
response_time_threshold(*?resp*), *?resp*$_x$<=*?resp*.

However, characterizing a trustee's behavior good or bad does not necessarily mean that this trustee will be added to the truster's white or black list, respectively. This decision is left to the truster's private strategy and it could vary greatly from agent to agent. For instance, a truster could be lenient and, thus, it might add quite easily trustees to its whitelist. Another truster might expect to see good behavior several times either for the same reason (r_{11}, where *?self* represents the truster itself) or for a number of reasons (r_{12}), at least four different, before adding a trustee to its whitelist. Similarly, a truster might expect to face a trustee's bad behavior more than one times either for the same reason (r_{13}) or for a number of reasons (r_{14}), before adding the trustee to its blacklist. Hence, a strict truster would easily add trustees to its blacklist but not to its whitelist whereas a lenient would give more chances before adding a trustee to its own blacklist. Notice that, since old good / bad behaviors are not deleted from the system, we define a time_window variable in order to avoid praising / penalizing an agent for ever (where *now()* returns the current time point).

r_{11}: *add_whitelist*(*trustee*→ *?x*, *time* → *?t3*) :=
time_window(?wtime),
good_behavior(*time* → *?t1*, *truster*→ *?self*, *trustee*→ *?x*, *reason* → *?r*),
good_behavior(*time* → *?t2*, *truster*→ *?self*, *trustee*→ *?x*, *reason* → *?r*),
good_behavior(*time* → *?t3*, *truster*→ *?self*, *trustee*→ *?x*, *reason* → *?r*),
?t3 > ?t2 > ?t1 >= now() - ?wtime.

r_{12}: *add_whitelist*(*trustee*→ *?x*, *time* → *?t3*) :=
time_window(?wtime),
good_behavior(*time* → *?t1*, *truster*→ *?self*, *trustee*→ *?x*, *reason* → *?r1*),
good_behavior(*time* → *?t2*, *truster*→ *?self*, *trustee*→ *?x*, *reason* → *?r2*),
good_behavior(*time* → *?t3*, *truster*→ *?self*, *trustee*→ *?x*, *reason* → *?r3*),
good_behavior(*time* → *?t3*, *truster*→ *?self*, *trustee*→ *?x*, *reason* → *?r4*),
?t3 > ?t2 > ?t1 >= now() - ?wtime.
?r1 ≠ ?r2, *?r1 ≠ ?r3*, *?r2 ≠ ?r3*, *?r4 ≠ ?r3*, *?r4 ≠ ?r2*, *?r4 ≠ ?r1*.

r_{13}: *add_blacklist*(*trustee*→ *?x*, *time* → *?t2*) :=
time_window(?wtime),
bad_behavior(*time* → *?t1*, *truster*→ *?self*, *trustee*→ *?x*, *reason* → *?r*),
bad_behavior(*time* → *?t2*, *truster*→ *?self*, *trustee*→ *?x*, *reason* → *?r*),

?t2 > ?t1 >= now() - ?wtime.

r_{14}: *add_blacklist*(*trustee*→ *?x, time* → *?t3*) :=

time_window(?wtime),

bad_behavior(*time* → *?t1, truster*→ *?self, trustee*→ *?x, reason* → *?r1*),

bad_behavior(*time* → *?t2, truster*→ *?self, trustee*→ *?x, reason* → *?r2*),

bad_behavior(*time* → *?t3, truster*→ *?self, trustee*→ *?x, reason* → *?r3*),

?t3 > ?t2 > ?t1 >= now() - ?wtime, ?r2 ≠ ?r1, ?r3 ≠ ?r2, ?r3 ≠ ?r1.

Mention that the above rules are defeasible since they are part of the truster's preferences (private strategy). The priority relationship among them could vary from case to case and it is left to the truster. Given that an agent can show good behavior in some dimensions and bad behavior in some other dimensions, at the same time (e.g. being correct but slow in response), both an add_whitelist and an add_blacklist conclusion can be inferred at the same time. In order to avoid conflicting list inclusions, we act skeptically by having the following defeaters which block such conflicting conclusions.

¬*add_whitelist*(*trustee*→ *?x, time* → *?t*) :~

add_blackist(*trustee*→ *?x, time* → *?t*)

¬*add_blackist*(*trustee*→ *?x, time* → *?t*) :~

add_blackist_whitelist (*trustee*→ *?x, time* → *?t*)

Other theories could also be used, depending on the requirements and preferences a truster has, in particular its human user. Next, as soon as the truster decides upon who should be added to the whitelist and/or the blacklist, it proceeds to the next part of rules (r_{15} – r_{18}) where the addition is actually carried out.

r_{15}: *blacklist*(*trustee*→ *?x, time* → *?t*) :=

¬*whitelist*(*trustee*→ *?x, time* → *?t1*),

add_blacklist(*trustee*→ *?x, time* → *?t2*),

?t2 > ?t1.

r_{16}: ¬*blacklist*(*trustee*→ *?x, time* → *?t2*) :=

blacklist(*trustee*→ *?x, time* → *?t1*),

add_whitelist(*trustee*→ *?x, time* → *?t2*),

?t2 > ?t1.

r_{17}: *whitelist*(*trustee*→ *?x, time* → *?t*) :=

¬blacklist(*trustee*→ *?x, time* → *?t1*),

add_whitelist(*trustee*→ *?x, time* → *?t2*),

?t2 > ?t1.

r_{18}: ¬*whitelist*(*trustee*→ *?x, time* → *?t2*) :=

whitelist(*trustee*→ *?x, time* → *?t1*),

add_blacklist(*trustee*→ *?x, time* → *?t2*),

?t2 > ?t1.

Notice that according to the above theory, when an agent is in the blacklist and it is decided to be added to the whitelist, then the agent is simply deleted from the blacklist, but it is not included in the whitelist (r_{156}). This can be done in a later moment if a similar *add_whitelist* decision is made again (r_{17}). A similar behavior for agents moving from the whitelist to the blacklist is followed by the two complementary rules r_{18} and r_{15}. Thus, apart from the two lists, there is a third "neutral zone" list, which needs not be explicitly realized. Only agents in the neutral zone can move directly to the lists. Notice that agents in the neutral zone can be there due to three reasons: 1) moved out from the whitelist, 2) moved out from the blacklist, 3) never been either in the whitelist or the blacklist. In order to handle all the above cases, we need to augment rules r_{15}- r_{18} with the following 4 rules:

r'_{15}: *blacklist*(*trustee*→ *?x, time* → *?t2*) :=
¬blacklist(*trustee*→ *?x, time* → *?t1*),
add_blacklist(*trustee*→ *?x, time* → *?t2*),
?t2 > *?t1*.

r''_{15}: *blacklist*(*trustee*→ *?x, time* → *?t*) :=
add_blacklist(*trustee*→ *?x, time* → *?t*),
not(whitelist(*trustee*→ *?x*)).

r'_{17}: *whitelist*(*trustee*→ *?x, time* → *?t2*) :=
¬whitelist(*trustee*→ *?x, time* → *?t1*),
add_whitelist(*trustee*→ *?x, time* → *?t2*),
?t2 > *?t1*.

r''_{17}: *whitelist*(*trustee*→ *?x, time* → *?t*) :=
add_whitelist(*trustee*→ *?x, time* → *?t*),
not(blacklist(*trustee*→ *?x*)).

As a result, the truster's whitelist, $WL_A \equiv \{X_i, \ldots X_n\}$, finally, includes the names (?x → X_i) of all its favored agents ($r_{19} - r_{20}$) whereas its blacklist, $BL_A \equiv \{X_j, \ldots X_m\}$, includes the agents that it would prefer to avoid ($r_{21} - r_{22}$). Notice, also, that the WL and BL sets are disjoint.

r_{19}: *WL*(*trustee*→ *?x*) :=
whitelist(*trustee*→ *?x, time* → *?t1*),
not(*¬whitelist*(*trustee*→ *?x, time* → *?t2*), *?t2* > *?t1*)).

r_{20}: *¬WL*(*trustee*→ *?x*) :~
¬whitelist(*trustee*→ *?x, time* → *?t1*),
not(*whitelist*(*trustee*→ *?x, time* → *?t2*), *?t2* > *?t1*)).

r_{21}: *BL*(*trustee*→ *?x*) :=
blacklist(*trustee*→ *?x, time* → *?t1*),
not(*¬blacklist*(*trustee*→ *?x, time* → *?t2*), *?t2* > *?t1*)).

r_{22}: *¬BL*(*trustee*→ *?x*) :~
¬blacklist(*trustee*→ *?x, time* → *?t1*),

not(*blacklist*(*trustee*→ *?x, time* → *?t2*), *?t2* > *?t1*)).

3.3.2 Locating ratings

A major challenge for open distributed and sometimes large-scale (multi-agent) systems is how to locate ratings among the rest of the community. The simplest and most common approach in such a distributed environment is to send a request message [75, 39]. Yet, the question is how and to whom this message should be sent directly and probably propagated by the direct and indirect receivers. To this end, using as a guide research on peer-to-peer networks [2], there are two core ways to propagate messages in order to locate peers (or ratings in our case) [52]. The first approach assigns a maximum time-to-live (TTL) parameter to each request message hence the requesting peer sends the message to its neighbors, who relay it to their own neighbors and so on until the time-to-live value is reached. The second approach allows peers to relay the message only to one neighbor at time, since they have to wait the response from a neighbor before forwarding the message to another neighbor. The first approach increases the communication cost, leading to significant higher bandwidth consumption but partners (and so ratings) are located fast. On the other hand, the second approach requires low bandwidth but it leads to time delays since more time is required to get feedback for the requests.

Over the last years, a number of researchers have proposed approaches that try to reduce bandwidth or improve response time (e.g. [48, 59]), mainly focusing on how to reach good and far away peers. Although, it is out of the scope of this article to research or improve peer-to-peer message propagate protocols, we were inspired by these approaches [48, 59, 17]. To this end, DISARM, proposes a more intuitive approach where agents take advantage of their previously established relationships in order to propagate their new requests, finding, quite fast, ratings with small bandwidth cost. More specifically, although the notion of neighbors does not exist in MASs, agents can use previously known partners in a similar way. To this end, in DISARM MASs are considered as social networks of agents. Such a social network can actually be represented as a social graph; a graph based on previously known agents either good (whitelist) or bad (blacklist). Hence, the known agents of an agent are, in our point of view, its neighbors. Using the knowledge represented by the social graph, DISARM is able to determine the proximity relationships among agents in the environment. In this context, it is easier for an agent to propagate its requests and eventually locate appropriate ratings.

Hence, an agent A that wants to collect ratings referred to agent X, does not send a request message to all agents but only to those stored in its whitelist. The motivation behind this action is the fact that a user is much more likely to believe statements from a trusted acquaintance than from a previously known dishonest agent or a stranger. Of course, agents are not always honest but previously known and well behaved agents are more likely to be honest. In this context, DISARM tries to use as many rating sources as possible while, through its mechanism, it tries to eliminate dishonest and misleading ratings (subsections 3.3.3 and 3.3.6).

Yet, these previously known and well behaved agents may have no interaction history with agent X. This could lead to limited or zero feedback for the requesting agent A. To this end, adopting the notion of TTL, in DISARM each ratings request message is accompanied with a TTL value, where TTL represents the hops in the graph. Hence, each request is characterized by its horizon (TTL value) that determines how far the message will be propagated in the network. In other words, the requesting agent determines if it is allowed (TTL $\neq 0$) for its known (whitelisted agents) to ask their own known agents, namely agents included in their white lists ($WL \equiv \{X_k, \ldots X_l\}$) and so on. Hence, the request message will be propagated in steps; each time an agent receives

such a request forwards it to its well-behaved known agents, if it is allowed (TTL ≠ 0), reducing the TTL value by one. However, if the requesting agent is included in the blacklist then its request message is ignored.

Moreover, the TTL value acts as a termination condition so that messages are not propagated indefinitely in the MAS; whenever an agent receives a request message with zero TTL does not forward the message. Finally, each agent will return, following the reverse path of the request, both its ratings and those provided by its partners, which eventually will be received by the initial requesting agent A. The above rule-based framework is, actually, logic independent since it can be implement in any logic. Yet, in DISARM, we use defeasible logic, as already mentioned, for purposes of simplicity and efficiency.

Rules r_{23}-r_{27}, below, model the above mentioned behavior. More specifically, rule r_{23} initiates the ratings requests by sending it to all agents *?r* in the whitelist, along with the TTL parameter. Notice that the *locate_ratings* fact always has a single instance at a certain time point, i.e. the system retracts this information after the transaction with a certain agent is performed. Rule r_{24} is responsible for answering back to the requesting agent about the requested agent's rating if such a previous experience exists in the local knowledge base. Rule r_{25} is responsible for forwarding a received request to agents in the whitelist if the TTL is still positive, by decreasing it at the same time. Rule r_{26} is a defeater rule that defeats rules r_{25}, namely it will block answering back to bad agents. Finally, rule r_{27} will store in the local knowledge base received ratings, if the sender is not in the blacklist. Notice that in this rule we use negation as failure, meaning that if *BL*(*?s*) fails during execution then not(*BL*(*?s*)) will succeed in order to determine if a sender agent does not belong to the blacklist.

r_{23}: *send_message*(*sender*→*?self*, *receiver*→*?r*, *msg* →*request_reputation*(*about*→*?x*, *ttl*→*?t*)) :=
ttl_limit(*?t*),
WL(*?r*),
locate_ratings(*about*→*?x*).

r_{24}: *send_message*(*sender*→*?self*, *receiver*→*?s*,
msg →*rating*(*id*→ id_x, *truster*→ *?self*, *trustee*→ *?x*, *t*→*?t*,
response_time→$?resp_x$, *validity*→$?val_x$, *completeness*→$?com_x$,
correctness→$?cor_x$, *cooperation*→$?coop_x$, *outcome_feeling*→$?outf_x$,
confidence→$?conf_x$, *transaction_value*→$?trans_x$)) :=
receive_message(*sender*→*?s*, *receiver*→*?self*, *msg* →*request_rating*(*about*→*?x*,*ttl*→*?tl*)),
rating(*id*→$?id_x$, *truster*→ *?self*, *trustee*→ *?x*, *t*→*?t*,
response_time→$?resp_x$, *validity*→$?val_x$, *completeness*→$?com_x$,
correctness→$?cor_x$, *cooperation*→$?coop_x$, *outcome_feeling*→$?outf_x$,
confidence→$?conf_x$, *transaction_value*→$?trans_x$).

r_{25}: *send_message*(*sender*→*?s*, *receiver*→*?r*, *msg* →*request_reputation*(*about*→*?x*,*ttl*→*?t1*)):=
receive_message(*sender*→*?s*, *receiver*→*?self*, *msg* →*request_rating*(*about*→*?x*,*ttl*→*?t*)),
?t >0,
WL(*?r*),
?t1 is ?t - 1.

r_{26}: ¬*send_message*(*sender*→*?self, receiver*→*?s, msg* →*?m)* :~
send_message(*sender*→*?self, receiver*→*?s, msg* →*?m),*
BL(*?s*).

r_{27}: *rating*(*id*→*?id*$_x$, *truster*→ *?x, trustee*→ *?y, t*→*?t,*
response_time→*?resp*$_x$, *validity*→*?val*$_x$, *completeness*→*?com*$_x$,
correctness→*?cor*$_x$, *cooperation*→*?coop*$_x$, *outcome_feeling*→*?outf*$_x$,
confidence→*?conf*$_x$, *transaction_value*→*?trans*$_x$) :=
receive_message(*sender*→*?s, receiver*→*?self,*
msg →*rating(id*→*?id*$_x$, *truster*→ *?x, trustee*→ *?y, t*→*?t,*
response_time→*?resp*$_x$, *validity*→*?val*$_x$, *completeness*→*?com*$_x$,
correctness→*?cor*$_x$, *cooperation*→*?coop*$_x$, *outcome_feeling*→*?outf*$_x$,
confidence→*?conf*$_x$, *transaction_value*→*?trans*$_x$)),
not(*BL*(*?s*)).

3.3.3 Discarding ratings

As soon as all available ratings are collected, an important decision has to be made; which ratings will be taken into account. Since agents may be dishonest, providing misleading ratings, DISARM attempts to minimize the effect of that dishonesty. To this end, one of the main goals of DISARM is to let agents (trusters) to use as many rating categories as possible. More specifically, ratings represent the experience of the involved parties, which is distinguished to direct (agent's direct experience PR_X) and indirect experience. Indirect experience is divided in two categories, ratings provided by strangers (SR_X) and reports provided by known agents due to previous interactions. In this context, r_{28} determines which agents are considered as *known*. Additionally to that, in DISARM known agents are divided to three more categories; agents included in the WL whitelist (WR_X), agents included in the BL blacklist (BR_X) and the rest known agents (KR_X). It is well known that using different opinions of a large group maximizes the possibility of crossing out unfair ratings. Hence, using both direct and indirect experience could lead to more truthful estimations.

r_{28}: *known*(*agent*→*?x*) :-
rating(*id*→*?id*$_x$, *truster*→*?self, trustee*→*?x*).

Sometimes one or more rating categories are missing, for instance, a newcomer has no personal experience and, thus, there are no available ratings (PR_X). To this end, we wish to ground our conclusions in trust relationships that have been built and maintained over time, much as individuals do in real world. For instance, a user is much more likely to believe statements from a trusted acquaintance than from a stranger. Thus, personal opinion (PR) is more valuable than acquaintances opinion (KR), which in turn is more valuable than strangers' opinion (SR). Moreover, previously known and blacklisted agents are generally considered unreliable compared to trusted agents (known agents or agents in the whitelist) and, thus, they are ignored. Finally, agents in the whitelist (WR) are usually more trusted than mere acquaintances (KR). In this context, the relationships among the rating categories are presented graphically in Figure 1. The root level includes the four available categories, each next level presents the combinations that can be derived from the (category) sets of its lower level. The edges present this relation for each case.

In order to understand Figure 1, the first level (top) suggests that all ratings count equally, whereas the fourth line (bottom), suggests an absolute preference to personal experience (PR), over whitelisted acquaintances (WR), over mere acquaintances (KR), and finally over strangers (SR). Thus, nodes on the left have precedence over nodes on the right. Furthermore, combinations of nodes from different levels can be made, provided that each rating source (PR, WR, KR, SR) is included only once. For example, one can combine node {PR, WR} from the third level with nodes {KR}, {SR} from the bottom level. This means that personal experience and experience of absolutely trusted acquaintances is treated equally, and both of them are preferred over ratings from mere acquaintances and over ratings from strangers.

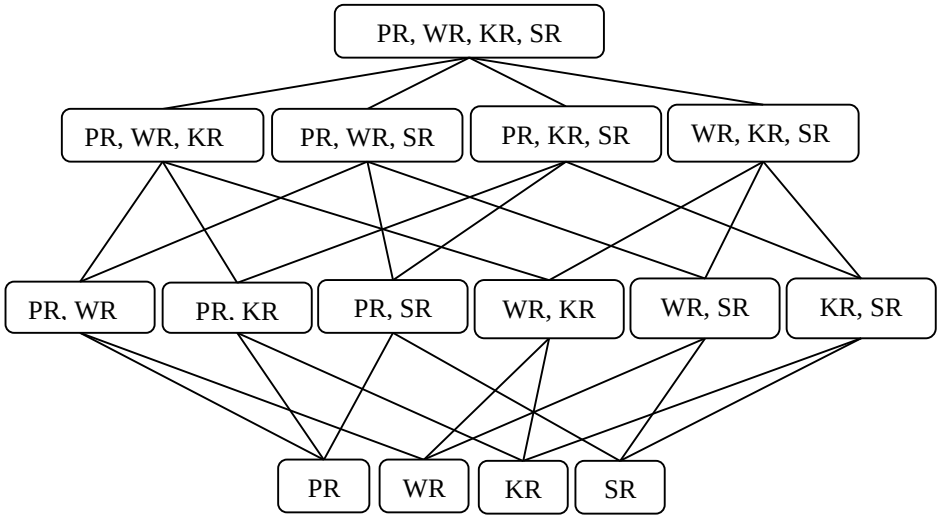

Fig. 1 Superiority relationships among rating categories.

As soon as the requesting agent A collects all the available ratings, it has to decide upon which of them will participate in the estimation. In order to do this, it has first to indicate which of them are eligible for participating in the final reputation value of agent X; namely a combination of four coefficients: $R_X = \{PR_X, WR_X, KR_X, SR_X\}$. Hence, as already discussed, DISARM uses confidence and transaction values in order to help agents discard some of the collected ratings. Besides, it is important to take into account ratings that were made by confident trusters, since their ratings are more likely to be right. Additionally, confident trusters, that were interacting in an important for them transaction, are even more likely to report truthful ratings. This assumption led to the following defeasible rules that define which ratings will be eligible for the reputation estimation and which not, according to the confidence and the transaction values, yet confidence and importance values are not involved in the estimation itself.

r_{29}: *eligible_rating*(*rating*→*?id*$_x$, *cat*→*1*, *truster*→*?a*, *trustee*→ *?x*) :=
confidence_threshold(*?conf*),
transaction_value_threshold(*?tran*),
rating(*id*→*?id*$_x$, *truster*→*?a*, *trustee*→ *?x*,
confidence→*?conf*$_x$, *transaction_value*→*?tran*$_x$),
?conf$_x$ >= *?conf*, *?tran*$_x$ >= *?tran*.

r_{30}: *eligible_rating*(*rating*→*?id*$_x$, *cat*→*2*, *truster*→*?a*, *trustee*→ *?x*) :=

confidence_threshold(*?conf*),

transaction_value_threshold(*?tran*),

rating(*id*→*?id*$_x$, *truster*→*?a*, *trustee*→ *?x*,

confidence→*?conf*$_x$, *transaction_value*→*?tran*$_x$),

?conf$_x$ >= *?conf*.

r_{31}: *eligible_rating*(*rating*→*?id*$_x$, *cat*→*3*, *truster*→*?a*, *trustee*→ *?x*) :=

confidence_threshold(*?conf*),

transaction_value_threshold(*?tran*),

rating(*id*→*?id*$_x$, *truster*→*?a*, *trustee*→ *?x*,

confidence→*?conf*$_x$, *transaction_value*→*?tran*$_x$),

?tran$_x$ >= *?tran*.

$r_{29} > r_{30} > r_{31}$

To this end, rule r_{29} indicates that if both the truster's confidence and transaction importance are high, according to the user's threshold, then that rating will be eligible for the estimation process. Rule r_{30}, on the other hand, indicates that even if the transaction value is lower than the threshold, it doesn't matter so much if the truster's confidence is high. Rule r_{31}, finally, indicates that if there are only ratings with high transaction value then they should be eligible. In any other case, the rating should be omitted. Notice that the above rules are defeasible and they all conclude positive literals. However, these literals are conflicting each other, for the same pair of agents (truster and trustee), since we want in the presence e.g. of personal experience to omit strangers' ratings. That's why there is also a superiority relationship between the rules. The conflict set is formally determined as follows:

C[*eligible_rating*(rating→?idx, *cat*→*?c*, *truster*→*?a*, *trustee*→*?x*)] =

{ ¬*eligible_rating*(rating→?idx, *cat*→*?c*, *truster*→*?a*, *trustee*→*?x*) } ∪

{ *eligible_rating*(rating→?idx, *cat*→*?c1*, *truster*→*?a*, *trustee*→*?x*) | *?c1* ≠ *?c* }

Moreover, even if it is defined which ratings are eligible, the final choice is up to the requesting agent A's personal strategy. The criterion for this final choice, as already mentioned, is time. Other agents will prefer to take into account all eligible ratings whereas others will move one step further indicating which of the eligible ratings, e.g. the newest, will finally participate in the estimation. For instance, rules r_{32}, $r_{32'}$ and $r_{32''}$ are examples of such a decision; r_{32} indicates that only ratings in a given time interval will count, $r_{32'}$ indicates that only the latest ratings (from a specific time point onwards) will count, whereas $r_{32''}$ indicates that only ratings reported back to a time window will count (where *now()* returns the current time point).

r_{32}: *count_rating*(*rating*→*?id*$_x$, *truster*→*?a*, *trustee*→ *?x*) :=

time_from_threshold(?ftime),

time_to_threshold(*?ttime*),

rating(*id*→*?id*$_x$, *t*→*?t*$_x$, *truster*→*?a*, *trustee*→ *?x*),

?ftime <= *?t*$_x$ <= *?ttime*.

r_{32}': *count_rating*(*rating*→*?id*$_x$, *truster*→*?a*, *trustee*→ *?x*) :=
time_from_threshold(*?ftime*),
rating(*id*→*?id*$_x$, *t*→*?t*$_x$, *truster*→*?a*, *trustee*→ *?x*),
?ftime <= *?t*$_x$.

r_{32}'': *count_rating*(*rating*→*?id*$_x$, *truster*→*?a*, *trustee*→ *?x*) :=
time_window(*?wtime*),
rating(*id*→*?id*$_x$, *t*→*?t*$_x$, *truster*→*?a*, *trustee*→ *?x*),
now() - ?*wtime* <= *?t*$_x$.

DISARM takes all the above into account and eventually classifies the ratings (rules r_{33} to r_{36}) into the previously defined categories (PR, WR, KR, SR) and takes the final decision about which of the ratings can actually participate in the estimation process for the final reputation value R_X (rules r_{37} to r_{39}).

r_{33}: *count_pr* (*rating*→*?id*$_x$, *trustee*→*?x*) :-
eligible_rating(*rating* → *?id*$_x$, *cat*→*?c*, *truster*→*?self*, *trustee*→ *?x*),
count_rating(*rating* → *?id*$_x$, *truster*→*?self*, *trustee*→ *?x*).

r_{34}: *count_wr* (*rating* →*?id*$_x$, *trustee*→*?x*) :-
known(*agent*→*?k*),
WL (*trustee* →*?k*),
eligible_rating(*rating* → *?id*$_x$, *cat*→*?c*, *truster*→*?k*, *trustee*→ *?x*),
count_rating(*rating*→*?id*$_x$, *truster*→*?k*, *trustee*→ *?x*).

r_{35}: *count_kr* (*rating* →*?id*$_x$, *trustee*→*?x*) :-
known(*agent*→*?k*),
not(*BL* (*trustee* →*?k*)),
not(*WL* (*trustee* →*?k*)),
eligible_rating(*rating*→*?id*$_x$, *cat*→*?c*, *truster*→*?k*, *trustee*→ *?x*),
count_rating(*rating*→*?id*$_x$, *truster*→*?k*, *trustee*→ *?x*).

r_{36}: *count_sr* (*rating*→*? id*$_x$, *trustee*→*?x*) :-
eligible_rating(*rating* → *?id*$_x$, *cat*→*?c*, *truster* →*?s*, *trustee*→ *?x*),
count_rating(*rating* → *?id*$_x$, *truster* →*?s*, *trustee*→ *?x*),
not(*known*(*agent*→*?s*)).

Rules r_{33} to r_{36}, classify the counted ratings in PR_X (direct experience), WR_X (known and trusted / whitelisted witness), KR_X (just known witness) and SR_X (strangers' witness), respectively. In r_{35} and r_{36}, we use negation as failure. Specifically, in r_{35} if an agent *?k* cannot be found both in the whitelist and the blacklist, it is considered as a *known* witness. Furthermore, in r_{36} if *known*() fails during execution then not(*known*()) will succeed, in order to determine which agents are considered totally strangers. Notice that the above rules are strict ones, i.e. their conclusions cannot be disputed.

The final decision making process for the R_X is based on a relationship theory among the rating categories. In Figure 1, we presented the complete relationship lattice among all rating categories, whereas, below, we present three potential theories based on that relationship lattice. In the first theory, all categories (attribute *rt*) count, hence, if ratings from all of them are available (rules r_{37} to r_{40}), then they will all participate in the final reputation estimation. To this end, if one of them is missing, then the other two are combined, whereas if just one category is available, then just this one will only be taken into account. This theory is equivalent to the first row in Figure 1, namely the combination {PR, WR, KR, SR}.

r_{37}: *participate*(*rating*→*?id*$_{AX}$, *rt*→*p*, *trustee*→*?x*) :=
count_pr(*rating*→ *?id*$_{AX}$, *trustee*→*?x*).

r_{38}: *participate*(*rating*→*?id*$_{WX}$, *rt*→*w*, *trustee*→*?x*) :=
count_wr(*rating*→ *?id*$_{WX}$, *trustee*→*?x*).

r_{39}: *participate*(*rating*→*?id*$_{KX}$, *rt*→*k*, *trustee*→*?x*) :=
count_kr(*rating*→ *?id*$_{KX}$, *trustee*→*?x*).

r_{40}: *participate*(*rating*→*?id*$_{SX}$, *rt*→*s*, *trustee*→*?x*) :=
count_sr(*rating*→ *?id*$_{SX}$, *trustee*→*?x*).

In the rest two theories, opinions from different categories conflict each other (conflicting literals), therefore the conflict is being resolved via adding superiority relationships. Specifically, personal opinion is the most important, and then comes whitelisted agents' opinion, then simply known agents' and then strangers'. We will present only the superiority relationships and we will not duplicate the rules. The conflict set (for both theories) is:

C[*participate*(*rating*→*?id*, *rt*→?*type*, *trustee*→?*x*)] =
{ ¬ *participate*(*rating*→*?id*, *rt*→?*type*, *trustee*→?*x*) } ∪
{ *participate*(*rating*→*?id*, rt→?*type1*, *trustee*→?*x*) | ?*type* ≠ ?*type1* }

In the second theory, the priority relationship among the rules is based on the fact that an agent relies on its own experience if it believes it is sufficient, if not it acquires the opinions of others, much as do humans in real life. This theory is equivalent to the last row in Figure 1, namely the combination {PR}, {WR}, {KR}, {SR}.

$r_{37} > r_{38} > r_{39} > r_{40}$

In the third theory, on the other hand, if direct experience is available (PR), then it is preferred to be combined with ratings from well trusted agents (WR). On the other hand, if personal experience is not available, then ratings from well trusted agents is preferred over just known agents, which is preferred over ratings coming from strangers. In the end, if nothing of the above is available, DISARM acts as a pure witness system. This theory is equivalent to the combination of the first node of the third row in Figure 1, with the two last nodes of the last row, namely the combination {PR, WR}, {KR}, {SR}.

$r_{37} > r_{40}$, $r_{37} > r_{39}$, $r_{38} > r_{39}$, $r_{38} > r_{40}$, $r_{39} > r_{40}$

3.3.4 Estimating Reputation

Agent A eventually reaches on a decision upon which rating is going to participate in the estimation ($R_X = \{PR_X, WR_X, KR_X, SR_X\}$), according to the chosen relationship theory, as discussed above. In this context, in order to cross out outliers, extremely positive or extremely negative values, the rating values are logarithmically transformed. Outliers are rating values that differ significantly from the mean (a central tendency) and, thus, they can have a large impact on the estimation process that could mislead agents. To this end, the most important feature of the logarithm is that, relatively, it moves big values closer together while it moves small values farther apart and, thus, rating data are better analyzed. More specifically, many statistical techniques work better with data that are single-peaked and symmetric while it is easier to describe the relationship between variables when it is approximately linear. Thus, when these conditions are not true in the original data, they can often be achieved by applying a logarithmic transformation.

To this end, each rating is normalized ($r \in [-1,1] \mid -1 \equiv$terrible, $1 \equiv$perfect), by using 10 as the logarithm base. Thus, the final reputation value ranges from -1 to +1, where -1, +1, 0 stand for absolutely negative, absolutely positive and neutral, respectively, which means that an agent's reputation could be either negative or positive. Hence, the final reputation value R_X is a function $\Im$ that combines the transformed ratings for each available category:

$$R_X = \Im\left(PR_x, WR_x, KR_x, SR_x\right) (1)$$

Moreover, since DISARM aims at simulating human behavior, it allows agents to determine what and how important is each rating parameter for them. In other words, an agent may consider validity more important than all, while it may not care at all about the outcome feeling of the interaction. An example could be the following: {*response time→20%, validity→50%, completeness→10%, correctness→10%, cooperation→10%, outcome feeling→0%*}. Hence, agents are allowed to provide specific weights (w_i, $i \in [1, 6]$) that will indicate their personal preferences according the ratings' coefficients. Formula 2, which is the modified Formula 1, calculates the weighted normalized values:

$$R_X = \Im\left[\frac{AVG\left(w_i \times \log\left(pr_X^{coefficient}\right)\right)}{\sum_{i=1}^{6} w_i}, \frac{AVG\left(w_i \times \log\left(wr_X^{coefficient}\right)\right)}{\sum_{i=1}^{6} w_i}, \frac{AVG\left(w_i \times \log\left(kr_X^{coefficient}\right)\right)}{\sum_{i=1}^{6} w_i}, \frac{AVG\left(w_i \times \log\left(sr_X^{coefficient}\right)\right)}{\sum_{i=1}^{6} w_i}\right],$$

$$coefficient = \{response\_time, validity, completeness, correctness, cooperation, outcome\_feeling\} (2)$$

Moving one step further, we try to understand deeper the relationship among the rating categories that participate in the estimation. It is up to the chosen relationship theory, presented in the previous subsection, which categories will participate, yet there is no clue about their percentage use in the estimation. To this end, in DISARM the user, through his/her agent A, is able to set what we call the "social trust weights" (π_p, π_w, π_k, π_s). These weights specify the balance between personal experience (π_p) and witness reputation (π_w, π_k, π_s). Hence, the final reputation value R_X is calculated according to which experience is more important for the end user (Formula 3).

$$R_X = \Im\left[\begin{array}{l} \frac{\pi_p}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \log\left(pr_X^{coefficient}\right)\right)}{\sum_{i=1}^{6} w_i}, \frac{\pi_w}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \log\left(wr_X^{coefficient}\right)\right)}{\sum_{i=1}^{6} w_i}, \\ \frac{\pi_k}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \log\left(kr_X^{coefficient}\right)\right)}{\sum_{i=1}^{6} w_i}, \frac{\pi_s}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \log\left(sr_X^{coefficient}\right)\right)}{\sum_{i=1}^{6} w_i} \end{array}\right],$$

$$coefficient = \{response\_time,\ validity,\ completeness,\ correctness,\ cooperation,\ outcome\_\ feeling\} \quad (3)$$

Finally, since time is an important aspect in reputation, DISARM allows time not only to be used for discarding available ratings but also to be used in the estimation itself. It is generally accepted that more recent ratings "weigh" more since they represent the latest activity of a specific agent. In order to include this aspect in the final reputation value, each rating at time t ($t_{start} < t < t_{current}$) is multiplied with t itself, as shown below. So, time becomes a sort of weight; the larger (i.e. the most recent), the more it weighs.

$$R_X = \Im\left[\begin{array}{l} \frac{\pi_p}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \frac{\sum_{\forall t_{start}<t<t_{current}} [\log\left(pr_X^{coefficient}(t)\right) \times t]}{\sum_{\forall t_{start}<t<t_{current}} t}\right)}{\sum_{i=1}^{6} w_i}, \\ \frac{\pi_w}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \frac{\sum_{\forall t_{start}<t<t_{current}} [\log\left(wr_X^{coefficient}(t)\right) \times t]}{\sum_{\forall t_{start}<t<t_{current}} t}\right)}{\sum_{i=1}^{6} w_i}, \\ \frac{\pi_k}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \frac{\sum_{\forall t_{start}<t<t_{current}} [\log\left(kr_X^{coefficient}(t)\right) \times t]}{\sum_{\forall t_{start}<t<t_{current}} t}\right)}{\sum_{i=1}^{6} w_i}, \\ \frac{\pi_s}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \frac{\sum_{\forall t_{start}<t<t_{current}} [\log\left(sr_X^{coefficient}(t)\right) \times t]}{\sum_{\forall t_{start}<t<t_{current}} t}\right)}{\sum_{i=1}^{6} w_i} \end{array}\right],$$

$$coefficient = \{response\_time,\ validity,\ completeness,\ correctness,\ cooperation,\ outcome\_\ feeling\} \quad (4)$$

Hence, Formula 4 represents DISARM's final metric. Moreover, note that a potential example of this formula, the simplest one for function $\Im$, could be the summation; in the sense that all categories participate additively in the final value, each one with its own weight. Notice that any multi-criteria decision making method / function could be used instead. However, we believe that the weighted sum model is the most intuitive one.

For demonstration purposes regarding the formula (4), a small example case is presented below. Consider an agent (A) that wants to estimate the reputation value of another agent (X). It possess four (4) ratings, one (1) from its own experience, one (1) collected from a white-listed agent W (a believed-to-be honest and reliable agent), one (1) collected from another known agent K and, finally, one (1) collected from a total stranger S. Hence, each rating category includes just one rating since our intention is to demonstration how formula 4 is used. More ratings can be added without differentiating the estimation procedure. These ratings are presented below. Note that in order to eliminate complexity we assume that time represents the simulation round, hence it takes integer values ($t \in [1,10]$).

PR_X = { *rating*(*id*→*1*, *truster*→*A*, *trustee*→*X*, *t*→*2*, *response_time*→*9*, *validity*→*7*,

completeness→6, correctness→6, cooperation→8,

outcome_feeling→7, confidence→0.9, transaction_value→0.8)}

WR_X = { *rating*(*id→3, truster→*W, *trustee→X, t→*5, *response_time→7, validity→8,*
completeness→7, correctness→8, cooperation→9,
outcome_feeling→8, confidence→0.7, transaction_value→0.6)}

KR_X = { *rating*(*id→4, truster→K, trustee→X, t→3, response_time→7, validity→6,*
completeness→7, correctness→6, cooperation→7,
outcome_feeling→6, confidence→0.7, transaction_value→0.7)}

SR_X = {*rating*(*id→*8, *truster→S, trustee→X, t→8, response_time→9, validity→9,*
completeness→9, correctness→9, cooperation→8,
outcome_feeling→8, confidence→0.7, transaction_value→0.9)}

Furthermore, the agent A determines its preferences about both the so-called social trust weights (π_p, π_w, π_k, π_s) and the weights it associates to the rating parameters. More specifically, for this case response time counts 40%, validity counts 20%, completeness counts 10%, correctness counts 10%, cooperation counts 10% and outcome feeling counts also 10% while π_p counts 40%, π_w counts 30%, π_k counts 20%, π_s counts 10%. Using the above ratings, the weights and a normalized summation function for the formula (4), the final reputation value is estimated as:

$$R_X = \log\left(\frac{\pi_p}{\pi_p+\pi_w+\pi_k+\pi_s}\times\frac{AVG\left(w_i\times\frac{\sum_{\forall t_{start}<t<t_{current}}[\log(pr_X^{coefficient}(t))\times t]}{\sum_{\forall t_{start}<t<t_{current}} t}\right)}{\sum_{i=1}^{6} w_i}+\frac{\pi_w}{\pi_p+\pi_w+\pi_k+\pi_s}\times\frac{AVG\left(w_i\times\frac{\sum_{\forall t_{start}<t<t_{current}}[\log(wr_X^{coefficient}(t))\times t]}{\sum_{\forall t_{start}<t<t_{current}} t}\right)}{\sum_{i=1}^{6} w_i}+\right.$$
$$\left.\frac{\pi_k}{\pi_p+\pi_w+\pi_k+\pi_s}\times\frac{AVG\left(w_i\times\frac{\sum_{\forall t_{start}<t<t_{current}}[\log(kr_X^{coefficient}(t))\times t]}{\sum_{\forall t_{start}<t<t_{current}} t}\right)}{\sum_{i=1}^{6} w_i}+\frac{\pi_s}{\pi_p+\pi_w+\pi_k+\pi_s}\times\frac{AVG\left(w_i\times\frac{\sum_{\forall t_{start}<t<t_{current}}[\log(sr_X^{coefficient}(t))\times t]}{\sum_{\forall t_{start}<t<t_{current}} t}\right)}{\sum_{i=1}^{6} w_i}\right),$$

$$coefficient=\{response\_time,\ validity,\ completeness,\ correctness,\ cooperation,\ outcome\_feeling\}\quad(4)$$

$$R_x=\log\left(\frac{40}{40+30+20+10}\times\frac{AVG\left(40\times\frac{\log(pr_x^{response\_time}(2))\times2}{10},20\times\frac{\log(pr_x^{validity}(2))\times2}{10},10\times\frac{\log(pr_x^{completeness}(2))\times2}{10},10\times\frac{\log(pr_x^{correctness}(2))\times2}{10},10\times\frac{\log(pr_x^{cooperation}(2))\times2}{10},10\times\frac{\log(pr_x^{outcome\_feeling}(2))\times2}{10}\right)}{100}+\right.$$
$$\frac{30}{40+30+20+10}\times\frac{AVG\left(40\times\frac{\log(wr_x^{response\_time}(5))\times5}{10},20\times\frac{\log(wr_x^{validity}(5))\times5}{10},10\times\frac{\log(wr_x^{completeness}(5))\times5}{10},10\times\frac{\log(wr_x^{correctness}(5))\times5}{10},10\times\frac{\log(wr_x^{cooperation}(5))\times5}{10},10\times\frac{\log(wr_x^{outcome\_feeling}(5))\times5}{10}\right)}{100}+$$
$$\frac{20}{40+30+20+10}\times\frac{AVG\left(40\times\frac{\log(kr_x^{response\_time}(3))\times3}{10},20\times\frac{\log(kr_x^{validity}(3))\times3}{10},10\times\frac{\log(kr_x^{completeness}(3))\times3}{10},10\times\frac{\log(kr_x^{correctness}(3))\times3}{10},10\times\frac{\log(kr_x^{cooperation}(3))\times3}{10},10\times\frac{\log(kr_x^{outcome\_feeling}(3))\times3}{10}\right)}{100}+$$
$$\left.\frac{10}{40+30+20+10}\times\frac{AVG\left(40\times\frac{\log(sr_x^{response\_time}(8))\times8}{10},20\times\frac{\log(sr_x^{validity}(8))\times8}{10},10\times\frac{\log(sr_x^{completeness}(8))\times8}{10},10\times\frac{\log(sr_x^{correctness}(8))\times8}{10},10\times\frac{\log(sr_x^{cooperation}(8))\times8}{10},10\times\frac{\log(sr_x^{outcome\_feeling}(8))\times8}{10}\right)}{100}\right)$$

$$=\log\left(\frac{40}{100}\times\frac{AVG\left(40\times\frac{\log(9)\times2}{10},20\times\frac{\log(7)\times2}{10},10\times\frac{\log(6)\times2}{10},10\times\frac{\log(6)\times2}{10},10\times\frac{\log(8)\times2}{10},10\times\frac{\log(7)\times2}{10}\right)}{100}+\right.$$
$$\frac{30}{100}\times\frac{AVG\left(40\times\frac{\log(7)\times5}{10},20\times\frac{\log(8)\times5}{10},10\times\frac{\log(7)\times5}{10},10\times\frac{\log(8)\times5}{10},10\times\frac{\log(9)\times5}{10},10\times\frac{\log(8)\times5}{10}\right)}{100}+$$
$$\frac{20}{100}\times\frac{AVG\left(40\times\frac{\log(7)\times3}{10},20\times\frac{\log(6)\times3}{10},10\times\frac{\log(7)\times3}{10},10\times\frac{\log(6)\times3}{10},10\times\frac{\log(7)\times3}{10},10\times\frac{\log(6)\times3}{10}\right)}{100}+$$
$$\left.\frac{10}{100}\times\frac{AVG\left(40\times\frac{\log(9)\times8}{10},20\times\frac{\log(9)\times8}{10},10\times\frac{\log(9)\times8}{10},10\times\frac{\log(9)\times8}{10},10\times\frac{\log(8)\times8}{10},10\times\frac{\log(8)\times8}{10}\right)}{100}\right)$$

$$= \log\left(\frac{40}{100} \times \frac{AVG\left(40 \times \frac{0.95 \times 2}{10}, 20 \times \frac{0.84 \times 2}{10}, 10 \times \frac{0.77 \times 2}{10}, 10 \times \frac{0.77 \times 2}{10}, 10 \times \frac{0.9 \times 2}{10}, 10 \times \frac{0.84 \times 2}{10}\right)}{100} + \right.$$

$$\frac{30}{100} \times \frac{AVG\left(40 \times \frac{0.84 \times 5}{10}, 20 \times \frac{0.9 \times 5}{10}, 10 \times \frac{0.84 \times 5}{10}, 10 \times \frac{0.9 \times 5}{10}, 10 \times \frac{0.95 \times 5}{10}, 10 \times \frac{0.9 \times 5}{10}\right)}{100} +$$

$$\frac{20}{100} \times \frac{AVG\left(40 \times \frac{0.84 \times 3}{10}, 20 \times \frac{0.77 \times 3}{10}, 10 \times \frac{0.84 \times 3}{10}, 10 \times \frac{0.77 \times 3}{10}, 10 \times \frac{0.84 \times 3}{10}, 10 \times \frac{0.77 \times 3}{10}\right)}{100} +$$

$$\left. \frac{10}{100} \times \frac{AVG\left(40 \times \frac{0.95 \times 8}{10}, 20 \times \frac{0.95 \times 8}{10}, 10 \times \frac{0.95 \times 8}{10}, 10 \times \frac{0.95 \times 8}{10}, 10 \times \frac{0.9 \times 8}{10}, 10 \times \frac{0.9 \times 8}{10}\right)}{100} \right)$$

$$= \log\left(\frac{40}{100} \times \frac{AVG(7.6, 3.36, 1.54, 1.54, 1.8, 1.68)}{100} + \frac{30}{100} \times \frac{AVG(16.8, 9, 4.2, 4.5, 7.75, 4.5)}{100} + \frac{20}{100} \times \frac{AVG(10.08, 4.62, 2.52, 2.31, 2.52, 2.31)}{100} + \frac{10}{100} \times \frac{AVG(30.4, 15.2, 7.6, 7.6, 7.2, 7.2)}{100}\right)$$

$$= \log\left(\frac{40 \times 2.92}{100} + \frac{30 \times 7.79}{100} + \frac{20 \times 4.06}{100} + \frac{10 \times 12.53}{100}\right) = \log(1.168 + 2.337 + 0.812 + 1.253) = \log(5.57) = 0.74$$

The final reputation is a positive value ($R_x = 0.74$) which means that the agent X is expected to behave well in a potential future transaction.

3.3.5 Measuring estimation confidence

As already mentioned, DISARM also studies the variability of the ratings that were finally taken into account as a measure about the confidence of the estimation itself. Generally speaking, DISARM attempts to encourage honest behavior by propagating request messages to whitelisted agents, yet agents are not always honest, hence some rating will be misleading. For this purpose, we use standard deviation. It measures the amount of variation or dispersion from the average. Yet, in addition to expressing the variability of a population, the standard deviation is commonly used to measure confidence in statistical conclusions. In other words, the standard deviation is a measure of how spread out numbers are. A low standard deviation indicates that the data points (here ratings) tend to be very close to the mean, the expected value, hence it is more likely the estimation to be closer to the agent's actual behavior whereas it is more possible to have mainly honest ratings. On the other hand, a high standard deviation indicates that the data points (ratings) are spread out over a large range of values and, thus, it is difficult to predict the agent's behavior, which means that possibly a great amount of ratings is misleading.

In this context, formula 5 presents the standard deviation metric used in DISARM, where N represents the total amount of used (in formula 4) ratings (r).

$$\sigma = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(r_X^{coefficient} - \mu(r_X^{coefficient}))^2}, r_X^{coefficient} \in pr_X^{coefficient} \cup wr_X^{coefficient} \cup kr_X^{coefficient} \cup sr_X^{coefficient},$$

$$coefficient = \{response\_time,\ validity,\ completeness,\ correctness,\ cooperation,\ outcome\_feeling\} \quad (5)$$

More specifically, in DISARM the above formula does not participate in the estimation process itself nor affect, in any way, the reputation value. Its role is to act complement to DISARM's final metric (formula 4), in order to indicate the probability the estimated value (formula 4) to be close to reality. Furthermore, it indicates how honest (or not) were the requested for ratings agents, since a high probability implies quite honest ratings. The motivation behind the use of the standard deviation formula (formula 5) was the fact that although reputation models provide an estimated reputation value they do not provide any clue about how likely is this estimation to reflect the agent's true behavior. To this end, DISARM provides an additional tool (formula 5) in order to assist agents, and thus their users, to make the best for them choices.

3.3.6 Facing dishonesty

Finally, let's assume that the truster (A) estimates the reputation value of the trustee (X), using a set of ratings, and makes its choice. The question is if that decision has eventually proven correct or wrong. In other words, the question is if the truster has made a satisfying choice or not. In this context, each time a transaction between A and X is not satisfying, agent A has to consider why that happened. A first answer could be that agent X acted unexpectedly, although it was highly rated (high reputation value), due to a temporal fault. Actually, this is the case sometimes. Yet, in competitive environments like multi-agent systems, the cause is usually dishonesty. As already discussed, DISARM includes a request message propagation to whitelisted agents, since these agents are believed to be more honest. Yet, sometimes they are not. As a result, agent A receives misleading ratings that leads it to wrong assessments.

The standard deviation metric, presented in the previous section, is an early warning. Its goal is to estimate the confidence in reputation estimations. A low standard deviation peals the alarm; on the other hand, a high deviation does not. Hence, the truster (agent A) has, at least, to protect itself from any similar wrong doing in the future by judging its rating providers. To this end, rule r_{41}, presented below, indicates that if the standard deviation value, at a specific time point (?t) for the trustee X (?x) is lower than the agent A's associated threshold then the estimation is considered bad (*bad_assessment*). Of course, this holds as far as it concerns a specific agent that provided the rating(s), namely it acted as truster (?y). Note that in the following example, r_{41}, we assume that agent ?y acted as truster providing just a rating ($?id_x$) to agent A.

$r_{41:}$ *bad_assessment* (*time* → *?t*, *truster*→ *?y*, *trustee*→ *?x*) :-

rating(*id*→*?id*$_x$, *time*→*?t*$_x$, *truster*→*?y*, *trustee*→ *?x*),

standard_deviation(*value*→?*stdev*$_x$, *time* → *?t*, *trustee*→ *?x*),

standard_deviation_threshold(*?stdev*),

?stdev$_x$ < *?stdev*, *?t*$_x$<=*?t*.

In this context, DISARM provides a set of rules that rearranges agents with possible dishonest behavior in agent A's repositories (including white and black list). Of course, it depends on agent's personal strategy how tolerant it is. For instance, an agent that was misled may add all involved rating providers (?y) to its blacklist whereas another may expect an agent to be involved many times in such a bad assessment. Sometimes, an agent may just remove a rating provider from its whitelist either instantly or after a number of cases.

Below, we present one such possible theory; any other theory would be possible based on the agent's personal preferences. Similarly to section 3.3.1 (rules r_{11}-r_{14}) we use a time window for considering only recent bad assessments. Rule r_{42}, below, indicates that if there is an agent ?y that interacted with the current agent in the past and agent ?y provided two ratings to the current agent about a third agent ?x, a potential partner for the current agent, but these two assessments were not good, then the current agent decides to add agent ?y in its blacklist. In the same spirit, rule r_{43} indicates that if a rating provider is involved in a bad assessment three times for three different agents, it should be added to the blacklist.

r_{42}: *add_blacklist*(*trustee*→ *?y*, *time* → *?t2*) :=

time_window(*?wtime*),

bad_assessment(*time* → *?t1*, *truster*→ *?y*, *trustee*→ *?x*),

bad_assessment(*time* → *?t2*, *truster*→ *?y*, *trustee*→ *?x*),

?t2 > ?t1 >= now() - ?*wtime.*

r_{43}: *add_blacklist*(*trustee* → *?y*, *time* → *?t2*) :=

time_window(*?wtime*),

bad_assessment(time → *?t1, truster*→ *?y, trustee*→ *?x1),*

bad_assessment(time → *?t2, truster*→ *?y, trustee*→ *?x2),*

bad_assessment(time → *?t3, truster*→ *?y, trustee*→ *?x3),*

?t3 > ?t2 > ?t1 >= now() - ?*wtime, ?x2* ≠ *?x1*, *?x3* ≠ *?x2*, *?x3* ≠ *?x1*.

Notice that these rules are combined with rules r_{11} to r_{14} in section 3.3.1, to form the total update strategy of the two lists. Also notice that even if the above rules seems quite harsh (because they immediately add the rating agent in the blacklist), in practice they are not. Recall that when an agent is already in the whitelist, an add_blacklist conclusion just moves it out of the whitelist to the "neutral zone". It takes a second consecutive add_blacklist conclusion to put the agent in the blacklist (rules r_{15} to r_{18} in section 3.3.1).

3.4 Complexity analysis

In order to have a rough estimation of the complexity of the proposed methodology, with respect to the size of the problem, namely the number of agents N and the total number of time points T, we will follow two different approaches. The first one is based on the assumption that we can use a propositional defeasible rule engine, such as SPINdle [47], in order to implement the defeasible theory for calculating the ratings that should be used for estimating the reputation of an agent. According to [49], propositional defeasible logic has linear complexity to the number of literals. Our theory, however, is a first-order theory. Rules with free variables can be interpreted as rule schemas, that is, as the set of all variable-free instances. Therefore, in order to estimate the complexity of our defeasible theory we can estimate how many variable-free literal instances, i.e. facts, are generated (in the worst case) from our theory, including of course, the initial facts.

The most important set of facts is the ratings provided by each agent to another agent, i.e. predicate ratings/12. This predicate can have at most N*(N-1)*T instances, i.e. all agents have provided ratings for any other agent in the system, at all time points. There are no other facts in the defeasible theory that are influenced by the size of the problem, therefore the number of initial facts is $O(N^2T)$.

Now, let's examine the rules. Rules about establishing good / bad behavior (r_1-r_{10}) can introduce at most 2*K*N*N*T literals, where K is the number of different reasons that an agent can be characterized as good or bad and each agent can be characterized by any other agent at any time point (in the worst case). Parameter K does not depend on the size of the problem, but includes every possible combination of the D rating dimensions, namely $K=2^D$. In the case of our theory, D=8; therefore K=256. Since K is not scalable, we conclude that rules about establishing good / bad behavior introduce $O(N^2T)$ literals.

Rules about adding an agent to the white or blacklist (r_{11}-r_{14}) introduce (worst case) for each truster N*T literals, since add_blacklist and add_whitelist mutually exclude each other, due to the mutual defeaters. Thus, in total, again $O(N^2T)$ literals are introduced. Rules about moving agents in/out of the white and black lists (r_{15}-r_{18}) also introduce (in the worst case) $O(N^2T)$ literals, in total, since they have the same parameters like the previous ones. Rules that compile the two lists (r_{19}-

r_{22}) introduce (in the worst case) for each truster N literals, since the two lists are mutually excluded. Thus, in total, these rule introduce $O(N^2)$ literals.

Rules about locating ratings from other agents (r_{23}-r_{26}) introduce for each truster (2+T)*N*N literals, since every agent can send to any other agent (in the worst case, when TTL equals the total number of nodes in the network) a message asking for ratings about a third agent, which in the worst case can be any other agent in the system. Factor 2+T is due to the fact that a message can be an initial request, a request propagation or an answer to a request, which can include at worst ratings about an agent for all time points. Thus totally (for all agents) these rules introduce $O(N^3T)$ literals. Rule r26 does not introduce new literals; it just copies ratings from one agent to another.

Rules for selecting ratings that will participate in the trust reputation (r_{28}-r_{40}) introduce, in the worst case, the same number of literals with the initial set of ratings, namely $O(N^2T)$ literals. Rules about measuring agent dishonesty and deciding about including agents in the white or back lists, have the same complexity with the rules about good / bad behavior and list management, i.e. they also can introduce at worse $O(N^2T)$ literals.

If we sum up all the previous, we end up that the size of the factbase is $O(N^2T + N^2 + N^3T) \approx O(N^3T)$. Therefore, the time complexity of our theory, if we assume a propositional implementation, is $O(N^3T)$. However, notice that this complexity is due to the fact that we have assumed a shared memory for all agents. In the case that each agent implements its own local knowledge base, the defeasible theory will run independently at each agent locally and therefore the complexity will fall to $O(N^2T)$ for each agent.

Given that our current implementation does not use a propositional defeasible logic rule engine, but DR-Device [7] which is a first-order defeasible logic rule engine that is implemented on top of the CLIPS production rule engine [23], we can estimate our methodology complexity as follows. CLIPS uses the RETE algorithm [26] for incrementally matching facts in the factbase to rule conditions. Its computational complexity is O(RFP), where R is the number of rules in the rulebase, F is the number of facts in the factbase, and P is the average number of patterns / conditions in the condition of the rules.

In our theory, the size of the defeasible rulebase is independent of the size of the problem (43 rules). Each rule in the defeasible theory is translated into a fixed set of production rules (for details see [6, 7]), therefore the size of the production rulebase is also independent of the size of the problem. The same is true for the average number of patterns / conditions in the condition of the rules. Actually, in the worst case (rule r_1) rules have <10 condition patterns. Therefore, the only quantity that depends on the problem size is the factbase, which was shown previously to be $O(N^3T)$, in the case of a centralized rule engine, or $O(N^2T)$, in the case of decentralized rule engines at each agent, which is a more likely case. Therefore, using a different approach we can estimate again a similar complexity for our methodology, namely linear to time and quadratic to the number of agents.

4 Evaluation

For evaluation purposes, regarding DISARM, we combined two, quite popular, testbed environments, adopted from [37, 36], previously used in [44] and [41]. These testbeds are quite similar, just with slight differences in number of participants and simulation rounds and they were used in a number of evaluation cases. The initial testbed as well as its slight variation that is adopted

in this article were developed by a well-known research team for evaluation purposes regarding reputation models. This testbed without loss of generality reduces the complexity of the environment. Furthermore, it allows quickly obtainable and easily reproducible results. The foremost advantage of the testbed is the fact that it provides a realistic view of a multi-agent system's performance under commonly appeared conditions, such as realistic network latency, congestion, user behavior and so forth. In this context, we preserved the testbed design but slightly changed the evaluation settings, taking into account the data provided in previous works. Below, a description of the testbed and the experimental settings for our experiments are given. Next, the implementation methodology of DISARM and the evaluation results are also presented.

4.1 Testbed

The testbed environment is a multi-agent system consisting of agents providing services and agents that use these services, namely providers and consumers. We assume that the performance of a provider, and effectively its trustworthiness, in a specific service is independent from other services that it might offer. Hence, in order to reduce the complexity of the testbed's environment, without loss of generality, we assume that the performance of a provider is independent from the service that is provided. In this context, it is assumed that there is only one type of service in the testbed and, as a result, all the providers offer the same service.

Nevertheless, the performance of the providers, such as the quality of the service in terms of satisfaction, response time, etc., differs and determines the utility that a consumer gains from each interaction (called UG≡utility gain). The value of UG varies from 0 to 10 and it depends on the level of performance of the provider in that interaction. A provider agent can serve many users at a time. After an interaction, the consumer agent rates the service of the provider based on the level of performance and the quality of the service it received. Each agent interaction is a simulation round. Events that take place in the same round are considered simultaneous and, thus, the round number is used as the timestamp for events and ratings.

To this end, taking all the above into account, the testbed in each experiment is populated with provider and consumer agents. Each consumer agent is equipped with a particular trust model (a centralized approach is also included), which helps it select a provider when it needs to use a service. We assume that consumers select always the provider with the highest reputation value. Note that whenever there are no available ratings for a provider, its reputation value is zero. In this context, the only difference among consumer agents is the trust models that they use, so the utility gained by each agent through simulations will reflect the performance of its trust model in selecting reliable providers for interactions. As a result, the testbed records the UG of each interaction with each trust model used. Consumer agents without the ability to choose a trust model will randomly select a provider from the list. Furthermore, in order to obtain an accurate result for performance comparisons between trust models, each one is employed by a large but equal number of consumer agents.

Hence, Table 2 displays the testbed environment; the six trust models used by consumer agents (along with the absence of model) and the four types of service providers. Regarding trust models, the testbed includes DISARM (the proposed model), Social Regret [65], Certified Reputation [36], CRM [41], FIRE [37], HARM [29] and NONE (absence of trust mechanism). Social Regret takes into account a social dimension, attempting to heuristically reduce the number of queries that are required in order to locate ratings. For this purpose, it groups agents and asks the opinion of only an agent per group. Yet, this way Social Regret marginalizes most of the agents.

Certified Reputation, on the other hand, is a well-known model that asks agents to give ratings of their performance after every transaction while the agents that receive these ratings have to keep them. Hence, each agent that needs ratings is able to ask any other agent for its stored references. However, Certified Reputation is designed to determine the access rights of agents, rather than to determine their expected performance.

Table 2 Testbed environment.

Service Providers	***Population Density***	***Service Consumers (Trust model compliance)***	***Population Density***
Good providers	15%	DISARM	14%
Ordinary providers	30%	Social Regret [65]	14%
Bad providers	40%	Certified Reputation [36]	14%
Intermittent providers	15%	CRM [41]	14%
		FIRE [37]	14%
		HARM [44]	14%
		NONE	16%

CRM (Comprehensive Reputation Model) is a probability-based model that asks agents to keep ratings, both from their direct transactions and witnesses, calling the procedure online trust estimation. Later, the actual performance of an evaluated agent will be compared against the above related ratings, in order to judge the accuracy of the consulting agents in the previous on-line process. This procedure is called off-line. FIRE is a popular model that integrates interaction trust, role-based trust, witness reputation and certified reputation. All available values are combined into a single measure by using the weighted mean method. Finally, HARM, is a previous work of us that uses temporal defeasible logic in order to combine interaction trust and witness reputation. It is the only centralized approach that is taken into account since it is a hybrid rule-based reputation model that uses defeasible (yet temporal) logic from a similar viewpoint. Actually, DISARM is based on the principles of HARM, forming an updated and extended, distributed this time, model. More details for the above models can be found below in the Related Work section.

Regarding service providers the testbed includes good, ordinary, bad and intermittent providers, namely honest and malicious agents. The first three provide services according to the assigned mean value of quality with a small range of deviation. In other words, good, ordinary and bad providers have a mean level of performance, hence, their activity (actual performance) follows a normal distribution around this mean. Intermittent agents, on the other hand, cover all possible outcomes randomly (Table 3). More particularly, regarding their strategy, good providers act always honestly, providing immediately consumers with accurate and right services. Ordinary providers, on the other hand, are usually honest but they have sometimes a significant delay in response. Hence, ordinary provider agents respond always to a service call, providing usually the right service but most of the times with a delay. The above, good and ordinary providers, form the two honest cases. The testbed includes also two malicious cases, intermittent and bad providers. Intermittent providers respond usually without delay but most of the times the service they provide is not the expected but a wrong one. On the other hand, bad providers respond always with a delay, providing a wrong service. Bad providers are an obvious bad case, in the sense that they are absolutely malicious agents that provide dishonest services. As a result, they can be located quite easily with a well-formed reputation model. Yet, intermittent providers are a more complicated and dreaded case of malicious agents. They act immediately, providing either good or bad services,

without a specific behavior pattern. Hence, it is difficult for the rest agents to detect and reveal their malicious behavior. In this context, there is usually a vast amount of possibilities regarding the allocation of providers, yet, it is impossible to explore them exhaustively. Hence, we use a quite common case where just half of the providers lead to profit (satisfying UG value). Additionally, since good and intermittent providers are not so common in real life, these categories get a low percentage (15%) in the population.

Table 3 Providers' performance distribution.

Service Providers	***Mean performance (UG value)***	***Performance distribution (UG value)***
Good providers	9	[8, 10]
Ordinary providers	7	[6, 8]
Bad providers	3	[0, 6]
Intermittent providers	5	[0, 10]

Furthermore, since multi-agent systems are open systems, agents may join or leave the system at any time. In this context, in order to simulate this dynamic behavior, we remove a percentage of the testbed agents while we add new ones into it, at each simulation round. Yet, our intention is to maintain the different groups of categories and their relevant proportions. Hence, at each round just a 10% to 20% is renewed, by actually replacing agents in the system.

4.2 Complying with DISARM

For implementation purposes, we use EMERALD [45], a framework for interoperating knowledge-based intelligent agents in the Semantic Web. This framework is built on top of JADE [8], a reliable and widely used multi-agent framework. EMERALD was chosen since it provides a safe, generic, and reusable framework for modeling and monitoring agent communication and agreements. Moreover, it proposes, among others, a reusable prototype for knowledge-customizable agents (called KC-Agents) and the use of Reasoners [46]. The agent prototype promotes customizable agents, providing the necessary infrastructure for equipping them with a rule engine and a knowledge base (KB) that contains agent's knowledge and personal strategy. Complying with this prototype, agents that use DISARM have their ratings expressed as facts in RDF format (Figure 3) and their decision-making rules (defeasible logic) in RuleML [16] (Figure 2), a Semantic Web language. Hence, it is quite easy for these agents to "run" the DISARM model and reach a promising decision. Reasoners, on the other hand, are agents that offer reasoning services to the rest of the agent community. A Reasoner can launch an associated reasoning engine, in order to perform inference and provide results. EMERALD supports a number of Reasoners but most important for the purposes of this article are the four Reasoners that use defeasible reasoning; among them is the DR-Reasoner (based on DR-Device defeasible logic system [7]), the defeasible Reasoner that was used for the evaluation of DISARM.

Additionally, EMERALD provides an advanced yellow pages service, called AYPS, that is responsible for recording and representing information related to registered in the environment agents, namely their name, type, registration time and activity. This information is dynamically stored in the AYPS agent's database. Hence, the service is able to retrieve up-to-date information at any time. Hence, even if DISARM (or any other distributed model) is a distributed reputation model, agents that use it are able to send requests to AYPS in order to get first a list of potential

partners, which is the case for newcomers. Next, they will use the DISARM model in order to estimate reputation for one or more of them in order to find the most appropriate partner (higher reputation value). Of course, it is not necessary to use such services; it is up to each agent's personal strategy how it will locate potential partners. The more an agent knows the environment, the better it can choose providers and, thus, the more utility gains. In this context, agents in the environment are free to ask others for their opinion (ratings), hence each agent requests the service from the most trustworthy and reliable provider according to it.

```
<Rule>
  <Implies ruletype="defeasiblerule">
    <oid><Ind uri="r36">r36</Ind></oid>
    <head><Atom>
      <op> <Rel> participate </Rel></op>
      <slot>
        <Var> trustee </Var>
        <Var> rt </Var>
        <Var> rating </Var>
      </slot>
    </Atom></head>
    <body><Atom>
      <op><Rel> count_pr </Rel></op>
      <slot>
        <Var> trustee </Var>
        <Var> rating </Var>
      </slot>
    </Atom></body>
  </Implies>
</Rule>
```

Fig. 2 A defeasible rule (r_{36}) example in RuleML format.

```
<rdf:RDF>
  <disarm:rating rdf:about="&disarm_ex">
    <disarm:id rdf:datatype="&xsd;integer">1</disarm:id>
    <disarm:trustee>X</ disarm:trustee>
    <disarm:t>140630105632</disarm:t>
    <disarm:response_time>9</disarm:response_time>
    <disarm:validity >7</disarm:validity >
    <disarm:completeness>6</disarm:completeness>
    <disarm:correctness>6</disarm:correctness>
    <disarm:cooperation>9</disarm:cooperation>
    <disarm:outcome_feeling>7</disarm:outcome_feeling>
    <disarm:confidence>0.9</disarm:confidence>
    <disarm:transaction_value>0.9</ disarm:transaction_value>
  </ disarm:rating>
</rdf:RDF>
```

Fig. 3 A rating example in RDF format.

Furthermore, concerning DISARM's final metric (formula 4), in this section we adopt the summation aggregation function, as shown below:

$$R_X = \frac{\pi_p}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \frac{\sum_{\forall t<t_{current}} [\log\left(pr_X^{coefficient}(t)\right) \times t]}{\sum_{\forall t<t_{now}} t} \right)}{\sum_{i=1}^{6} w_i} + \frac{\pi_w}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \frac{\sum_{\forall t<t_{current}} [\log\left(wr_X^{coefficient}(t)\right) \times t]}{\sum_{\forall t<t_{now}} t} \right)}{\sum_{i=1}^{6} w_i} +$$

$$\frac{\pi_k}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \frac{\sum_{\forall t<t_{current}} [\log\left(kr_X^{coefficient}(t)\right) \times t]}{\sum_{\forall t<t_{now}} t} \right)}{\sum_{i=1}^{6} w_i} + \frac{\pi_s}{\pi_p+\pi_w+\pi_k+\pi_s} \times \frac{AVG\left(w_i \times \frac{\sum_{\forall t<t_{current}} [\log\left(sr_X^{coefficient}(t)\right) \times t]}{\sum_{\forall t<t_{now}} t} \right)}{\sum_{i=1}^{6} w_i},$$

$$coefficient = \{response\_time,\ validity,\ completeness,\ correctness,\ cooperation,\ outcome\_feeling\} \quad (4)$$

In this context, Figure 4 displays the overall transaction steps that follows an agent that complies to DISARM methodology. First, an agent A that wants to choose an appropriate provider requests ratings from its whitelisted agents (usually TTL≠0). If TTL is greater than zero, the request message is propagated to other agents in the environment (as described in section 3.3.2). Next, agent A will receive all available ratings. Then, it will ask for help from the DR-Reasoner in order to conduct inference on that data based on its personal preferences (its strategy). Having the results, it will eventually choose an agent X with the highest reputation value. As soon as, the transaction between agents A and X ends, agent A (truster) evaluates agent X (trustee), storing the rating for future use.

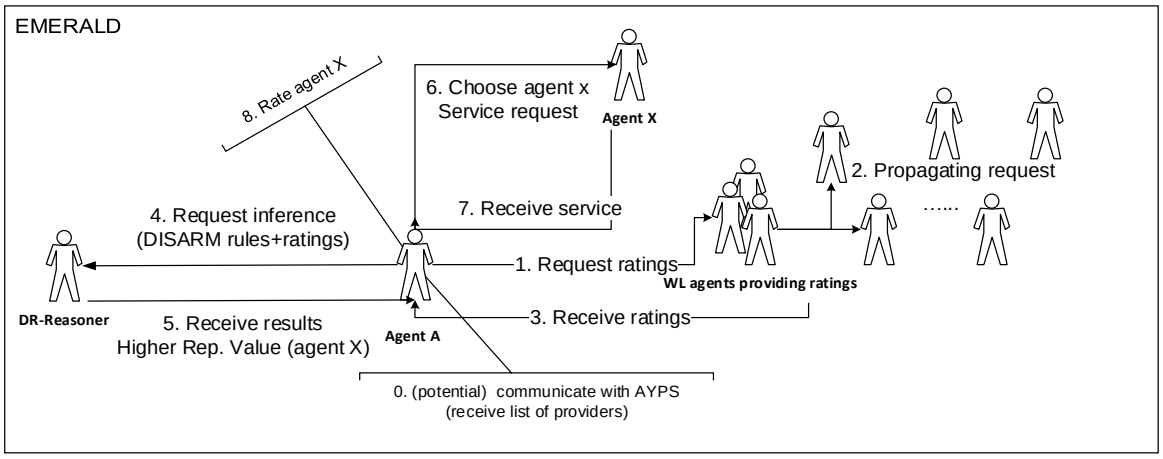

Fig. 4 DISARM transaction steps.

Finally, taking into account all the above we evaluated DISARM using the discussed settings. Yet, the first set of simulations was conducted just for DISARM. More specifically, we maintained all settings except the fact that we used only the DISARM model, hence all agent population was using the DISARM methodology (100%). Figure 5 displays the results. DISARM has an upward trend which indicates its ability to provide good estimations. Yet, it takes some time to reach good utility values which is not surprising since agents need time to interact and create relationships (known and whitelisted agents) in the network. Note that time (in Figures 5, 7 and 8) takes integer values since it is associated with the number of simulations. More specifically, each time point is associated with the respective simulation round.

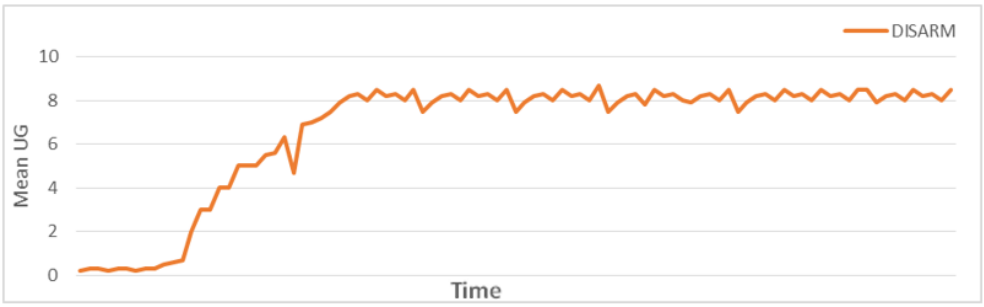

Fig. 5 Mean Utility Gained by DISARM over time.

In order to further evaluate the proposed model, we checked its behavior regarding dishonesty. In other words, since agents are not always honest, we study how DISARM captures different levels of dishonesty in the environment. Hence, we ran more simulation tests with different proportions of dishonest agents. Note that by dishonest agents, we refer to agents that provide dishonest rating reports to others. The starting density of dishonest agents in the population was 0% ending to 100% with a 10% step. Figure 6 displays the findings, namely the mean UG value for each case. DISARM achieves quite high UG values for most of the cases, even for a density of 60%, while it maintains a satisfying UG value even for extreme cases (70% - 100%). Beyond the rule-based decision mechanism of DISARM, significant proportion of this success is due to the use of standard deviation, presented in section 3.3.5. Agents complying with DISARM are able to evaluate the probability of the estimated reputation value to be close to reality, hence they can protect themselves from wrong choices. Besides, a typical behavior of a "DISARM" agent is to check first the standard deviation value; if this value is not satisfying the agent removes (if necessary) acquaintances from its whitelist, using defeasible theories such the ones presented in section 3.3.6) and rechecks trustee's (X) reputation value. Hence, it is able to make quite safe choices and receive, therefore, high UG values. Yet, if the density of dishonest agents is high (e.g. 70% - 100%) standard deviation is not able to detect it in a satisfactory manner. In other words, if almost everybody is lying it is difficult to detect the lie. It will be revealed afterwards. Yet, DISARM uses a variety of rating categories, hence, an agent that realizes that most agents in the environment are lying is able (based on its strategy) to use only its personal experience and/or reports from proved honest whitelisted agents. Hence, the system (an environment where agents are using DISARM) behaves smoothly, showing graceful degradation. In other words, the performance of the DISARM model and, as a result, the performance of the system falls smoothly related to the increase of dishonest agents density whereas there is no limit level beyond which the performance falls abruptly.

For purposes of better evaluating and understanding, Figure 6 displays also the results of another set of simulations. The test settings are the same except the fact that standard deviation is not used. Our aim is to compare these findings in order to discover the usefulness of standard deviation. In this context, the comparative study of these two graphs revealed that omitting standard deviation leads to lower UG values, approximately about 10% less. Hence, it is obvious that although the standard deviation metric is not used in the estimation process, it is an important element of the DISARM model.

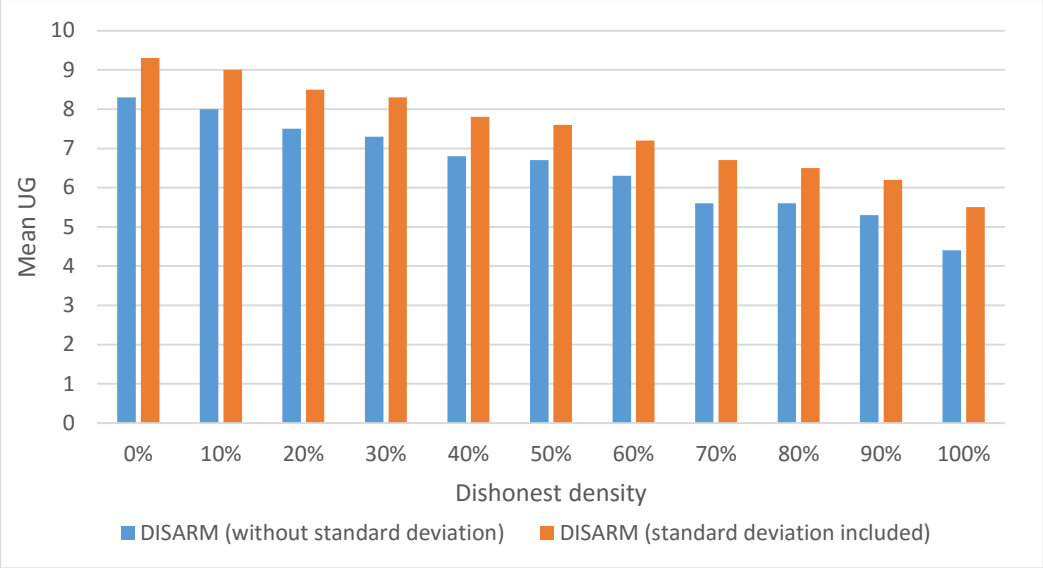

Fig. 6 Mean Utility Gained by DISARM in cases of dishonesty (standard deviation included).

4.3 Model Comparison

Finally following the above settings that form an environment with both honest and malicious agents, in this section, we compare DISARM with related trust models (see Table 2), such as Social Regret [65], Certified Reputation [36], CRM [41], FIRE [37], HARM [44] and NONE (no trust mechanism, randomly selected providers). We used HARM although it is a centralized approach since it is a rule-based model using temporal defeasible reasoning. In this context, taking into account all the available data, Figure 8 depicts the overall ranking regarding the mean utility gained (UG value) for all models, even for absence of model, namely NONE (random selection).

As shown in Figure 7 NONE performance is poor and, as expected, consistently the lowest. HARM, on the other hand, is consistently the highest. This is not surprising since HARM is a rule-based, centralized model. Hence, it is able to gather ratings about all interactions in the system as opposed to the rest distributed models, where locating rating is a challenging task by itself. This allows agents using HARM to achieve higher performance right from the first interactions. Concerning, the distributed models, it is clear that DISARM, Certified Reputation and Social Regret gain a quite high UG value, yet they are unable to reach the performance of centralized models like HARM. Among distributed models, DISARM achieves a slightly higher performance, probably mainly due to the fact that it is using a defeasible reputation estimation mechanism that enables agents to take decisions non-monotonically and, thus, are a better fir to a dynamic environment, increase agent's decision-making performance. Furthermore, DISARM enables agents to get familiarized with the environment faster as opposed to CRM and FIRE which, as shown in Figure 7, need more time to know the environment and stabilize their performance. Yet, Figure 7 compared to Figure 5 reveals that DISARM reaches higher performance when it is the only model used in the environment. A possible explanation for this is that in the first case there is a large percentage of agents that use it (100%), hence more social relationships are created. As a result, there is a larger amount of available ratings, leading to better rating quality and, thus, more accurate assessments.

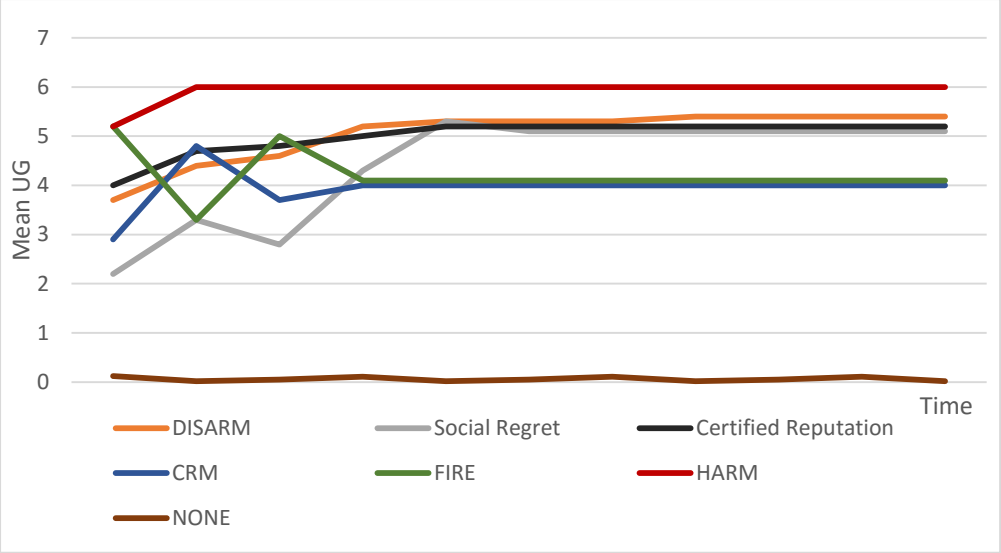

Fig. 7 Mean Utility Gained Ranking.

Finally, although (as shown in Figure 8), centralized models achieve higher UG score, they have significant limitations in terms of execution time and storage space. This is not surprising since centralized models are usually managed by a single agent. This manager has to store all ratings in the system, which are increased over time, and to respond to an increasing number of requests, leading to bottle-neck effect. On the other hand, distributed models store just their own ratings and those obtained by witnesses which are far less than the total number of available ratings in the system. Additionally, these extra witness ratings could be erased after use, releasing space. As shown in Figure 8, HARM, being a centralized model, needs much more space than distributed models, reaching even the double. On the other hand, models like DISARM and Social Regret that take into account social aspects, need less space, even from other distributed approaches, since they collect less ratings. Hence, the more ratings used by a model the more space (and usually time) is needed. Mention that, here, storage space stands for the percentage usage of the available space (e.g. MB) each agent has.

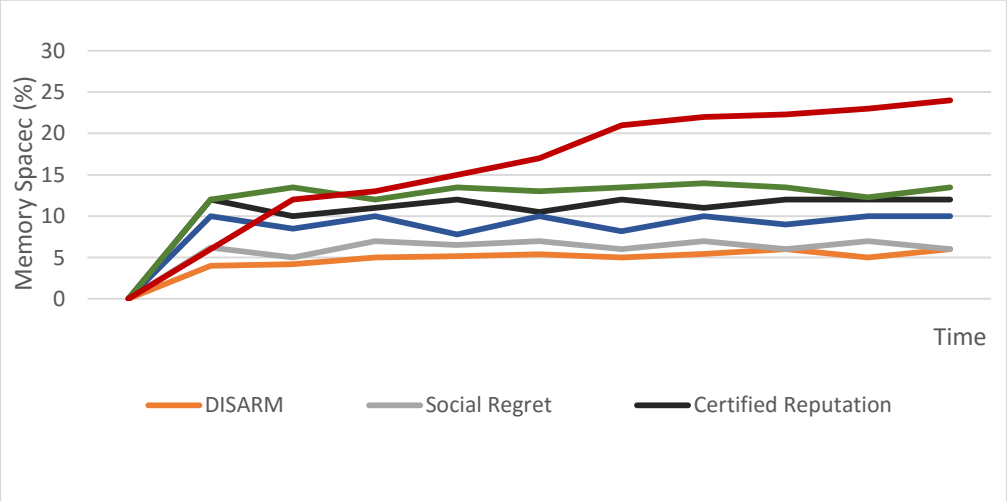

Fig. 8 Storage space growth.

5 Related Work

Trust and reputation represent a significant aspect in modern multi-agent systems. An interesting and very challenging active research area is already focused on them; various models and metrics have already been proposed in order to deal with the challenging decision making processes in the agent community [33, 30, 56]. Reputation is used to build trust among agents, minimizing the risk involved in the transactions and increasing users' confidence and satisfaction. Hence, since the best decisions are those that taken under the minimum risk, trust and reputation models support agents to take promising decisions regarding potential partners.

To this end, one of the first, if not the first, model that used the idea of witness reputation was a decentralized trust model, called Regret [64]. Regret is, actually, one of the most representative trust and reputation models in multi-agent systems. It combines witness reports and direct interaction experience in order to provide reputation values. Additionally, in Regret ratings are dealt with respect to time; old ratings are given less importance compared to new ones. An evolution of Regret, a primary attempt to locate witnesses' ratings, called Social Regret [65], was also presented by the authors. Social Regret is a reputation system oriented to e-commerce environments that incorporates the notion of social graph. More specifically, Social Regret groups agents with frequent interactions among them and considers each one of these groups as a single source of reputation values. In this context, only the most representative agent within each group is asked for information. To this end, a heuristic is used in order to find groups and to select the best agent to ask.

Social Regret, similarly to DISARM, is one of these cases that the social dimension of agents is taken into account. Yet, Social Regret does not reflect the actual social relations among agents, like DISARM, but rather attempts to heuristically reduce the number of queries to be done in order to locate ratings. Taking into account the opinion of only one agent of each group is a severe disadvantage since the most agents are marginalized, distorting reality. However, both Regret and DISARM recognize the importance of time and take into account both personal and witness ratings. Yet, only DISARM allows agents to decide on their own about what they consider important regarding time. Additionally, only DISARM provides a knowledge-based mechanism, promoting a nonmonotonic, more flexible, human-like approach.

Another popular distributed model is FIRE [37]. FIRE integrates four types of trust and reputation, namely interaction trust, role-based trust, witness reputation and certified reputation. Interaction trust and witness reputation are, as in DISARM, an agent's past experience from direct interactions and reports provided by witnesses about an agent's behavior, respectively. Role-based trust, on the other hand, is trust defined by various role-based relationships between the agents whereas certified reputation is third-party references provided by the target agents. The aforementioned values are combined into a single measure by using the weighted mean method. FIRE similar to DISARM recognizes the need for hybrid models that will take into account more than one source for the final reputation estimation. Yet, although FIRE take into account more sources than DISARM, it uses a weak computation model for the final combination and reputation estimation. DISARM, on the other hand, provides a human-like knowledge-based mechanism, based on defeasible logic that let agents take into account the most promising available rating in order to predict the future behavior of a potential partner. Furthermore, opposed to DISARM, FIRE does not take into consideration the problem of lying and inaccuracy. Additionally, only DISARM takes into account the social dimension of multi-agent systems with respect to time.

Another remarkable reputation model is Certified Reputation [36], a decentralized reputation model, like DISARM, involving each agent keeping a set of references given to it from other agents. In this model, each agent is asked to give certified ratings of its performance after every transaction. The agent then chooses the highest ratings and stores them as references. Any other agent can then ask for the stored references and calculate the agent's certified reputation. This model overcomes the problem of initial reliability in a similar way with DISARM. However, opposed to our approach, this model is designed to determine the access rights of agents, rather than to determine their expected performance. Furthermore, it is a witness-based model, whereas DISARM combines both witnesses and direct experience, providing a rule-based methodology to deal with the discrimination issue. Furthermore, although in Certified Reputation agents are freed from the various costs involved in locating witness reports, such as resource, time and communication costs, ratings might be misquoted since it is each agent's responsibility to provide ratings about itself and in the fact the best ones.

TRR (Trust–Reliability–Reputation) trust model [63] allows a software agent to represent both the reliability and the reputation of another agent, merging finally these measures into a global trust evaluation. It uses a dynamically computed weight that represents how an agent considers important the reliability with respect to the reputation when it computes the trust of another agent. Yet, the weight depends on the number of interactions between the two agents, which is actually a problem when these agents have no interaction history. Hence, TRR provides a mechanism for estimating the global reputation value of an agent based on previous interactions. On the other hand, DISARM provides a knowledge-based mechanism that enables each agent to estimate a personalized reputation based on its preferences. Moreover, DISARM takes into account plenty of issues, such as time and social relations, although it does not deal with reliability issues as TRR does. Additionally, DISARM is a nonmonotonic approach based on defeasible logic that provides a more flexible, human-like approach that enables agents not only to estimate a reputation value but also to decide upon their relationships in the community.

CRM (Comprehensive Reputation Model) [41] is another typical distributed reputation model. In CRM the ratings used to assess the trustworthiness of a particular agent can either be obtained from an agent's interaction history or collected from other agents that can provide their suggestions in the form of ratings; namely interaction trust and witness reputation, respectively. CRM is a probabilistic-based model, taking into account the number of interactions between agents, the timely relevance of provided information and the confidence of reporting agents on the provided data. More specifically, CRM, first, takes into account direct interactions among agents, calling the procedure online trust estimation. After a variable interval of time, the actual performance of the evaluated agent is compared against the information provided by other agents in a procedure called off-line. Off-line procedure considers the communicated information to judge the accuracy of the consulting agents in the previous on-line trust assessment process. In other words, in CRM the trust assessment procedure is composed of on-line and off-line evaluation processes. Both CRM and DISARM acknowledge the need for hybrid reputation models taking into account time issues, yet they propose a starkly opposite approach. Additionally, both models use a confidence parameter in order to weight ratings more accurately. However, DISARM takes into account a variety of additional parameters, allowing users to define weights about them. As a result, more accurate and personalized estimations are provided. Furthermore, only DISARM considers the social relations among agents providing a nonmonotonic approach that let them establish and maintain trust relationships, locating quite easily reliable ratings.

Finally, HARM [44], a previous work of us, is a hybrid rule-based reputation model that uses temporal defeasible logic in order to combine interaction trust and witness reputation. Yet, it is a centralized approach which actually overcomes the difficulty to locate witness reports. DISARM, on the other, hand is also a hybrid but distributed model that uses defeasible (yet not temporal) logic in a similar point of view. Actually, DISARM is an updated and extended model based slightly partially on HARM's basic principles though adapting a decentralized and social approach. To this end, although both models consider time important, they are dealing with it with a totally different approach. Ratings in HARM are characterized by a time offset property, which indicates the time instances that should pass in order to consider each rating active while each of them counts only for a limited time duration. DISARM uses the time itself in the final estimation formula, letting agents to use a similar to human thinking philosophy that first decides upon which category of rating should be taken into account and then discards ratings included there. Comparing, these two models, we believe that DISARM and HARM, despite their similarities and differences, are two nonmonotonic models that enable agents to improve their effectiveness and intuitiveness in a way more related to the traditional human reasoning for assessing trust in the physical word.

6 Conclusions and Future Work

This article presented DISARM, a social, distributed, hybrid, rule-based reputation model which uses defeasible logic. DISARM though appropriate rules, combines interaction trust and witness reputation. Moreover, it limits the common disadvantages of the existing distributed trust approaches, such as locating ratings, by considering the agents acting in the environment as a social network. Hence, each agent is able to propagate its requests to the rest of the agent community, locating quite fast ratings from previously known and well-rated agents. Additionally, DISARM's mechanism is based on defeasible logic, modeling the way intelligent agents, like humans, draw reasonable conclusions from inconclusive information, which is one of the main advantages of our approach. Actually, it is one of the first models that use nonmonotonic reasoning, in the form of defeasible logic in order to predict agents' future behavior. It is based on well-established estimation parameters [19, 20], such as information correctness, completeness, and validity, agent's response time and cooperation, as well as outcome feeling of the interaction. Hence, DISARM can be adopted in any multi-agent system in the Semantic Web, such as JADE and EMERALD. Finally, we provided an evaluation that illustrates the usability of the proposed model.

As for future directions, our intention is to study and address the issue and challenges that we have identified. To this end, first of all, we plan to study further DISARM's performance by comparing it to more reputation models from the literature and use it in real-world applications, combining it also with Semantic Web metadata for trust [21, 22]. More specifically, the Semantic Web acts as a decentralized system on which any human or artificial entity is able to act, using or publishing data. The trustworthiness of these data and entities most of the times is under question. Hence, trust models must adapt to these needs by taking into account additional parameters or technologies. For this purpose, the Semantic Web provides a useful tool, the so-called metadata. Metadata are useful for trust assessments with Semantic Web data, since they provide information about one or more aspects of the data they are associated, enable trust models to infer about the trustworthiness of Semantic Web data. In this context, we plan to study metadata and their challenges as well as the potential of DISARM to adopt this technology.

Another direction is towards further improving DISARM. There are still some open issues and challenges regarding, for instance, locating ratings. More technologies could be adopted for this purpose; ontologies, machine learning techniques and user identity recognition and management being some of them. Ontologies are considered one of the pillars of the Semantic Web, although they do not have a universally accepted definition. Usually, an ontology is considered as a formal specification of a shared conceptualization [31, 40]. In other words, an ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse. Over the years, ontologies have become common in the Web. This is not surprising since ontologies make domain assumptions explicit and clear, enabling information reuse and common understanding of the structure of information. Furthermore, ontologies have the ability to integrate existing ontologies describing portions of a large domain. Hence, formalizing reputation and, as a result trust, into ontologies has several advantages such as creating a common understanding for reputation and enabling mapping between reputation concepts. Moreover, ontologies increase the possibility of trust and reputation interoperability and cross community sharing of reputation information [1]. Therefore, we plan in the future to study how DISARM can benefit from ontologies. For this purpose, our intention is to develop or extend a reputation-oriented ontology that will let agents understand both each other and the model. Our ultimate goal is to strengthen the social dimension of our model, helping agents to locate partners and ratings more easily.

Another research area that could benefit trust and reputation models is machine learning, which has long been recognized for its parallelization and distribution. Sophisticated machine learning algorithms and methodologies, such as supervised learning, can be applied in order to analyze past transactions in an environment, modeling more precisely trust. Over the last years, a number of researchers proposed trust and reputation models that involve machine learning techniques, forming a new trend in the area. These attempts seem to reveal that machine learning methods, such as hierarchical and Bayesian learning, provide more accurate approaches in environments in which third party information is malicious, noisy, incomplete or inaccurate [70, 74]. Therefore, we plan to study a variety of machine learning methodologies in order to find out which of them can be integrated more efficiently in multi-agent systems. Our aim is to enrich DISARM with a powerful mechanism that will extract the relationships between potential partners as well as their past and future behavior.

Finally, since DISARM is a distributed approach based on defeasible logic, another direction for future work is to study if it can be implemented using the interesting distributed algorithms for Defeasible Logic described in [11]. This article argues about defeasible contextual reasoning with arguments in Ambient Intelligence, proposing an approach which is based on the MultiContext Systems (MCS) paradigm. In this approach, the involved entities are modeled as autonomous logic-based agents while their knowledge is encoded in rule theories. Furthermore, the associations between the context knowledge possessed by different ambient agents are modeled as mappings between their respective context theories. Missing or inaccurate knowledge is handled using defeasible logic. The above approach and DISARM, the proposed reputation model, feature many common principles and assumptions. Therefore, an attempt to implement DISARM using the above distributed algorithms would lead to valuable conclusions and perhaps an enriched implementation of DISARM.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers of the paper, whose comments helped to improve it a lot.

REFERENCES

1. Alnemr, R., & Meinel, C. (2011). From Reputation Models and Systems to Reputation Ontologies. Trust Management V, IFIP Advances in Information and Communication Technology, V 358, 98-116.
2. Androutsellis-Theotokis, S., & Spinellis, D. (2004). A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4): 335–371.
3. Antoniou, G., Billington, D., Governatori, G., & Maher, M. (2001). Representation results for defeasible logic. *ACM Trans. Comput. Log*. 2(2): 255-287
4. Antoniou, G., Maher, M., & Billington, D. (2000). Defeasible Logic versus Logic Programming without Negation as Failure. J. Log. Program. 42(1): 47-57.
5. Artz, D., & Gil, Y. (2007). A survey of trust in computer science and the Semantic Web. *Journal Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2): 58-71.
6. Bassiliades, N., & Vlahavas, I. (2006). R-DEVICE: An Object-Oriented Knowledge Base System for RDF Metadata. International Journal on Semantic Web and Information Systems, 2(2): 24-90.
7. Bassiliades, N., Antoniou, G., & Vlahavas, I. (2006). A Defeasible Logic Reasoner for the Semantic Web. *International Journal on Semantic Web and Information Systems*, 2(1):1-41.
8. Bellifemine, F., Caire, G., Poggi, A., & Rimassa, G. (2003). JADE: A white Paper. EXP in search of innovation, 3(3): 6-19.
9. Berners-Lee, T., Hall, W., Hendler, J., O'Hara, K., Shadbolt, N., & Weitzner, D. (2006) A Framework for Web Science. Foundations and Trends in Web Science, 1(1): 1–130.
10. Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American Magazine*, 284(5): 34-43. (Revised 2008)
11. Bikakis, A., & Antoniou, A. (2010). Defeasible Contextual Reasoning with Arguments in Ambient Intelligence. IEEE Trans. Knowl. Data Eng., 22(11): 1492-1506.
12. Billington, D. (1997). Conflicting literals and defeasible logic. Proceedings of 2nd Australian Workshop Commonsense Reasoning, 1-15.
13. Billington, D. (2008). Propositional Clausal Defeasible Logic. Logics in Artificial Intelligence, LNCS 5293: 34-47.
14. Billington, D., Antoniou, G., Governatori, G., Maher, M.J. (2010). An inclusion theorem for defeasible logics. Transactions on Computational Logic (TOCL), 12(1): 6.
15. Boley, H. (2010). Integrating Positional and Slotted Knowledge on the Semantic Web. *Journal of Emerging Technologies in Web Intelligence*, Pp. 343-353.
16. Boley, H., Paschke, A., & Shafiq, O. (2010). RuleML 1.0: The Overarching Specification of Web Rules. 4th Int. Web Rule Symposium: Research Based and Industry Focused (RuleML'10), Springer, 6403: 162-178.
17. Bosu, A., Carver, J., Guadagno, R., Bassett, B., McCallum, D., & Hochstein, L. (2014) Peer impressions in open source organizations: A survey. Journal of Systems and Software, Volume 94, August 2014, pp. 4-15.

18. Carrera, A., Iglesias, C., García-Algarra, J., & Kolařík, D. (2014). A real-life application of multi-agent systems for fault diagnosis in the provision of an Internet business service. *Journal of Network and Computer Applications*, 37:146-154.
19. Castelfranchi, C., & Falcone, R. (2010). *Trust Theory: A Socio-Cognitive and Computational Model*. Wiley Series in Agent Technology 1st edition, ISBN-13: 978-0470028759.
20. Castelfranchi, C., & Tan, YH. (2001). Trust and Deception in Virtual Societies. Springer, XXXI, 257.
21. Ceravolo, P., Damiani, E., & Viviani, M. (2006). Adding a Trust Layer to Semantic Web Metadata. In Soft Computing for Information Retrieval on the Web, 197: pp. 87-104.
22. Ceravolo, P., Damiani, E., & Viviani, M. (2007). Bottom-Up Extraction and Trust-Based Refinement of Ontology Metadata. IEEE Transactions on Knowledge and Data Engineering, 19(2):149-163.
23. CLIPS: A Tool for Building Expert Systems, http://clipsrules.sourceforge.net/.
24. Dasgupta. P. (2000). Trust as a commodity. Gambetta D. (Ed.). Trust: Making and Breaking Cooperative Relations, Blackwell, 49-72.
25. Ferretti, E., Errecalde, M., García, A., & Simari, G. (2007). An Application of Defeasible Logic Programming to Decision Making in a Robotic Environment. *Logic Programming and Nonmonotonic Reasoning*, Springer Berlin Heidelberg, LNCS, 4483: 297-302.
26. Forgy, C. L. (1991). Rete: a fast algorithm for the many pattern/many object pattern match problem. In Expert systems, Peter G. Raeth (Ed.). IEEE Computer Society Press, Los Alamitos, CA, USA 324-341.
27. Gottlob, G. (1992). Complexity Results for Nonmonotonic Logics. *Journal of Logic and Computation*, 2:397-425.
28. Governatori, G., & Rotolo, A. (2008). BIO Logical Agents: Norms, Beliefs, Intentions in Defeasible Logic. *Journal of Autonomous Agents and Multi_Agent Systems*, 17(1): 36-69.
29. Governatori, G., Hofstede, A., & Oaks, P. (2001) Is defeasible logic applicable?. Proceedings of the 2nd Australasian Workshop on Computational Logic, 47-62.
30. Grandison, T., & Sloman, M. (2000). A survey of trust in internet applications. IEEE Comm Surveys & tutorials, 3(4): 2-16.
31. Gruber. T. R. (1993). A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220.
32. Gutowska, A., (2008). Buckley, K. Computing Reputation Metric in Multi-agent E-commerce Reputation System. In 28th International Conference on Distributed Computing Systems, 255–260.
33. Han, Y., Zhiqi, S., Leung, C., Chunyan, M., & Lesser, VR. (2003). A Survey of Multi-Agent Trust Management Systems. Access, IEEE, l.1: pp. 35-50.
34. Hendler, J. (2001). Agents and the Semantic Web. *IEEE Intelligent Systems*, 16(2):30-37.
35. Hendrikx, F., Bubendorfer, K., & Chard, R. (2014). Reputation systems: A survey and taxonomy. *Journal of Parallel and Distributed Computing*, in press. DOI: 10.1016/j.jpdc.2014.08.004.
36. Huynh, D, Jennings, N R., & Shadbolt, N R. (2006). Certified reputation: How an agent can trust a stranger. In AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, Hokkaido, Japan.
37. Huynh, D., Jennings, N R., & Shadbolt, N. R. (2006). An integrated trust and reputation model for open multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems (AAMAS)*, 13(2):119-154.
38. Josang, A., Ismail, R., & Boyd, C. (2007). A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2): 618–644.
39. Jung, J. (2009). Trustworthy knowledge diffusion model based on risk discovery on peer-to-peer networks. *Expert Systems with Applications*, 36(3): 7123–7128.

40. Kashyap, V., Bussler, C., & Moran, M. (2008). *The Semantic Web, Semantics for Data and Services on the Web*. Springer-Verlag.
41. Khosravifar, B., Bentahar, J., Gomrokchi, M., & Alam, R. (2012). CRM: An efficient trust and reputation model for agent computing. *Kowledge-Based Systems*, 30:1-16.
42. Kontopoulos, E., Bassiliades, N., Antoniou, G. (2011). Visualizing Semantic Web proofs of defeasible logic in the DR-DEVICE system. *Knowledge-Based Systems*, 24(3):406-419.
43. Koons, R. (2009). Defeasible Reasoning. Stanford University: Stanford Encylopedia of Philosophy. Available at: http://plato.stanford.edu/entries/reasoning-defeasible/.
44. Kravari, K., & Bassiliades, N. (2012). HARM: A Hybrid Rule-based Agent Reputation Model based on Temporal Defeasible Logic. 6th International Symposium on Rules: Research Based and Industry Focused. Springer Berlin/Heidelberg, LNCS, 7438: 193-207.
45. Kravari, K., Kontopoulos, E., & Bassiliades, N. (2010). EMERALD: A Multi-Agent System for Knowledge-based Reasoning Interoperability in the Semantic Web. 6th Hellenic Conf. on Artificial Intelligence (SETN 2010). LNCS, 6040/2010: 173-182.
46. Kravari, K., Kontopoulos, E., & Bassiliades, N. (2010). Trusted Reasoning Services for Semantic Web Agents. *Informatica: International Journal of Computing and Informatics*, 34(4): 429-440.
47. Lam, H-P., & Governatori, G. (2009). The making of SPINdle. In Adrian Paschke, Guido Governatori, and John Hall, editors, Proceedings of The International RuleML Symposium on Rule Interchange and Applications (RuleML 2009). RuleML, Springer pp. 315-322.
48. Liao, Z., Liu, S., & Xi, S. (2014). An Improved Multipoint Relaying Scheme for Message Propagation in Distributed Peer-to-Peer System. *International Journal of Distributed Sensor Network*s, Article ID 792814, doi:10.1155/2014/792814.
49. Maher, MJ. (2001). Propositional defeasible logic has linear complexity. *Theory and Practice of Logic Programming*, 1(6):691–711.
50. Máhr, T.S., Srour, J., De Weerdt, M., & Zuidwijk, R. (2010). Can agents measure up? A comparative study of an agent-based and on-line optimization approach for a drayage problem with uncertainty. Transportation Research Part C: Emerging Technologies 18: 99.
51. Medić, A. (2012). Survey of Computer Trust and Reputation Models – The Literature Overview. *International Journal of Information and Communication Technology Research*, 2(3): 254-275.
52. Meshkova, E., Riihijärvi, J., Petrova, M., & Mähönen, P. (2008). A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Computer Networks*, 52(11): 2097–2128.
53. Nute, D. (1987). Defeasible Reasoning. 20th International Conference on Systems Science, IEEE, 470-477.
54. Nute, D. (1994). Defeasible logic. In Oxford University Press: Handbook of logic in artificial intelligence and logic programming: Nonmonotonic reasoning and uncertain reasoning, 3:353-395.
55. Nute, D. (2003). Defeasible Logic. In Theory, Implementation and Applications. Bartenstein O, Geske U, Hannebauer M, Yoshie O. (Eds.). Springer-Verlag: Web Knowledge Management and Decision Support: INAP 2001, 14th Int. Conference on Applications of Prolog (Revised Papers), 151-169.
56. Pinyol, I., & Sabater-Mir, J. (2013). Computational trust and reputation models for open multi-agent systems: a review. *Journal of Artificial Intelligence Review*. Springer Netherlands, 40(1): 1-25.
57. Pollock, J.L. (1998). Perceiving and Reasoning about a Changing World. *Computational Intelligence*, 14:498-562.
58. Pollock, JL. (1992). How to reason defeasibly. *Artificial Intelligence*. Elsevier, 57.
59. Ramanathan, M., Kalogeraki, V., Pruyne, J. (2002). Finding Good Peers in Peer-to-Peer Networks. International Parallel and Distributed Processing Symposium (IPDPS), 24-31.

60. Ramchurn, SD., Huynh, D., & Jennings, NR. (2004). Trust in multi-agent systems. *Knowledge Engineering Review*, 19(1):1-25.
61. Reiter, R. (1980). A Logic for Default Reasoning. *Artificial Intelligence Journal*, 13:81-132.
62. Resnick, P., Kuwabara, K., Zeckhauser, R., & Friedman, E. (2000). Reputation systems. *Communications of the ACM*, 43(12): 45-48.
63. Rosaci, D., Sarne, G., (2012). Garruzzo, S. Integrating Trust Measures in Multiagent Systems. *International Journal of Intelligent Systems*, 27: 1–15.
64. Sabater, J., & Sierra, C. (2001). Regret: A reputation model for gregarious societies. In Proceedings of the Fourth Workshop on Deception, Fraud and Trust in Agent Societies, 61-69.
65. Sabater, J., & Sierra, C. (2002). Social ReGreT, a reputation model based on social relations. SIGecom Exch, 3(1):44-56.
66. Sabri, K.E., & Obeid, N. (2015). A temporal defeasible logic for handling access control policies. *Journal of Applied Intelligence*, 1-13.
67. Sherchan, W., Loke, S., (2006) Krishnaswamy, S. A fuzzy model for reasoning about reputation in web services. In Proceedings of the 2006 ACM symposium on Applied computing (SAC '06), 1886–1892.
68. Skylogiannis, T., Antoniou, G., Bassiliades, N., Governatori, G., & Bikakis, A. (2007) DR-NEGOTIATE - A system for automated agent negotiation with defeasible logic-based strategies. *Data Knowledge Engineering*, 63(2): 362-380.
69. Su, X., Zhang, M., Mu, Y., & Bai, Q. (2011). Trust-based Service Provider Selection in Service-oriented Environments. IJCSNS International Journal of Computer Science and Network Security, 11(10): 1–9.
70. Teacy, L., Luck, M., Rogers, A., & Jennings, N. (2012). An efficient and versatile approach to trust and reputation using hierarchical Bayesian modelling. *Artificial Intelligence*, 193: 149-185.
71. Vrba, P., Radakovič, M., Obitko, M., & Mařík, V. (2011). Semantic technologies: latest advances in agent-based manufacturing control systems. *International Journal of Production Research*, 49(5): 1483-1496.
72. Wagner G. (2003). Web Rules Need Two Kinds of Negation. Proc. First Workshop on Semantic Web Reasoning, LNCS 2901, Springer 2003, pp. 33-50.
73. Xie, X.F., Smith, S., & Barlow, G. (2012). Schedule-driven coordination for real-time traffic network control. International Conference on Automated Planning and Scheduling (ICAPS), 323-331.
74. Xin, L., Gilles, T., & Anwitaman, D. (2014). A generic trust framework for large-scale open systems using machine learning. *Computational Intelligence*, 30(4): 1467-8640.
75. Yang, M., & Fei, Z. (2009). A novel approach to improving search efficiency in unstructured peer-to-peer networks. *Journal of Parallel and Distributed Computing*, 69(11): 877–884.
76. Zulkuf, G., Heidari, F., Oey, M., van Splunter, M., & Brazier, F. (2013). Agent-based information infrastructure for disaster management. *Intelligent Systems for Crisis Management*, 349–355.