HAL
open science

Reputation Put to the Test: a Testcase for a Cognitive Reputation Model

Juri Luca de Coi, Laurent Vercouter

To cite this version:

Juri Luca de Coi, Laurent Vercouter. Reputation Put to the Test: a Testcase for a Cognitive Reputation Model. 10th International Conference on Intelligent Agent Technologies (IAT), Aug 2011, Lyon, France. pp.248-255. hal-00641688

HAL Id: hal-00641688

https://hal.science/hal-00641688v1

Submitted on 16 Nov 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Reputation Put to the Test: a Testcase for a Cognitive Reputation Model

Juri Luca De Coi
Laboratoire Hubert Curien
Université Jean Monnet
Saint-Etienne, France
juri.luca.decoi@univ-st-etienne.fr

Laurent Vercouter
Laboratoire en Sciences et Technologies de l'Information
Ecole Nationale Supérieure des Mines
Saint-Etienne, France
laurent.vercouter@emse.fr

***Abstract*—The wide diffusion of open and decentralized environments like the Web makes it possible for actors to interact with previously unknown peers. As a consequence, trust has become a hot topic in the field of computer science. Many attempts to formalize concepts like *trust* and *reputation* have been carried out in the literature, most remarkably the one by Herzig et al. [5]. However, Herzig et al. focus on describing a conceptual framework but do not provide any concrete instantiation of it, thereby not showing any evidence about the effectiveness of their approach. This paper fills the gap by presenting such an instantiation based on agent technologies. Although our instantiation targets a Wikipedia-related scenario and exploits the Jason [1] and CArtAgO [13], [14] frameworks, the methodology we present is general and can be applied to different scenarios and agent technologies.**

***Keywords*-Trust; Reputation; Agent technology;**

I. Introduction

The wide diffusion of open and decentralized environments like the Web makes it possible for actors to interact with previously unknown peers. As a consequence, trust has become a hot topic in the field of computer science. Many attempts to formalize concepts like *trust* and *reputation* have been carried out in the literature [3], [10], [7]. For the context of digital societies, the lately presented approach by Herzig et al. [5] appears to be especially promising, since it goes beyond previous work by proposing a formal model that complies with the cognitive theories of social trust [3] and reputation [4].

However, Herzig et al. focus on describing a conceptual framework but do not provide any concrete implementation nor integration into software agents, thereby not showing any evidence about the effectiveness of their approach. This paper fills the gap by presenting such an integration targeting a meaningful real-world scenario related to Wikipedia.[1]

The conceptual framework by Herzig et al. nicely fits the Belief-Desire-Intention (BDI) model [2] commonly adopted in agent programming. For this reason, it was a natural choice resorting to agent technologies in order to implement the instantiation mentioned above. More specifically, we describe an implementation based on the Jason [1] and CArtAgO [13], [14] frameworks, although it must be clear that the methodology we present is general and can be applied to different agent technologies as well as to different scenarios.

The contributions of this paper are as follows.

- We extend the conceptual framework presented in [5] by introducing the notion of *reputation of an agent not to do an action*
- We present an instantiation of such conceptual framework targeting a meaningful real-world scenario related to Wikipedia
- We describe an implementation of such instantiation on top of Jason and CArtAgO

This paper is organized as follows: Section II describes the tackled scenario. Section III introduces the conceptual framework which will be used in the remaining of the paper. Section IV describes the instantiation of the conceptual framework targeting the scenario presented in Section II. Section V describes the implementation of the scenario. The outcome of an experimental evaluation is reported in Section VI. Section VII accounts for related work. Finally, Section VIII presents further work and draws some conclusions.

II. Running scenario

Because of the huge dimension of Wikipedia, "a number of Wikipedia community members have set up long-standing patrols [...] to help editors maintain reasonable quality on an encyclopedia this size [...] Patrols are used in Wikipedia to watch over a class of pages and take any appropriate actions".[2] Among Wikipedia patrols, the Recent changes patrol is responsible for checking "the recent changes to various articles for harmful *edits*" [12]. After identifying problematic edits, Recent changes patrollers (from now on *patrollers*) improve or revert the edits and possibly warn the *contributors* responsible for them.

[1] http://www.wikipedia.org/

[2] http://en.wikipedia.org/wiki/Wikipedia:Patrols

In order to simplify the activity of patrollers, a number of tools and resources are available (a brief account thereof is provided in Section V). However, such tools have been mainly designed to support the activity of patrollers who work independently from each other and do not allow for knowledge transfer among them. This approach has many drawbacks

- no patroller can profit from other patrollers' experience and lessons learned. As a consequence, a new-entry patroller will need for instance to check n edits of a contributor c before realizing that c is a vandal, even if another patroller already checked enough edits of c to conclude that she indeed is
- checking an edit's quality is not always trivial, since it often requires working knowledge about the topic of the edited article. Moreover—as confirmed by our investigation—simple and intuitive heuristics are not always effective (e.g., the use of a dirty or offensive word does not always indicate a vandalism, since it could be the only word known to a contributor not familiar with the article's language which has the intended meaning). The indication of the average quality of a contributor's edits as provided by other patrollers could save much time while evaluating upcoming edits of the same contributor
- tools available to patrollers often allow to add contributors to black- or whitelists. However, since checking an edit's quality is not trivial, patrollers could mistakenly add contributors to such lists. Again, the possibility of comparing a patroller's evaluations with other patrollers' ones could prevent or reduce the effect of such wrong additions
- a patroller might be a vandal herself. By sharing patrollers' evaluations and comparing them with each other, outlier behaviors (and hence vandal patrollers) can be easily identified

The arguments listed above motivate the need of knowledge transfer among patrollers. At a low abstraction level, such knowledge can be thought of as a set of statistics about the quality of the edits of different contributors. At a high abstraction level, such knowledge can be interpreted as an expectation about the quality of upcoming edits performed by the contributors, i.e., a sort of confidence that the upcoming edits of the contributors will have a given quality.

In the following sections, we show how the conceptual framework presented in [5] allows to express and represent such knowledge.

III. Conceptual framework

The scenario described in Section II requires patrollers to exchange confidence estimates about the quality of contributors' upcoming edits. In other words, it requires to build a common belief within a patroller community about the future behavior of contributors. Within [5]'s conceptual framework, such a belief can be most naturally modeled by resorting to the concept of reputation.

This section recalls the conceptual framework presented in [5] and proposes an extension by introducing the notion of *reputation of an agent not to do an action*.

A. Dispositional trust and reputation

The concepts of trust and reputation have been formalized in [5] according to the socio-cognitive approach presented in [3]. In this theory, trust is defined as follows: "an agent i trusts an agent j for the action α in order to achieve a goal φ". Two concepts defined in the formalization are especially interesting for our work: dispositional trust and reputation.

Dispositional trust refers to a trust belief occuring whenever some conditions are satisfied. Agent i is disposed to trust agent j to do α w.r.t. φ in the circumstances κ iff

- i has the potential goal φ in the circumstances κ
- i believes that from now on, if it has the goal φ and κ holds, then
  - j will be capable to do α
  - j, by doing α, will ensure φ to be true at some point
  - j will intend to do α

This definition describes dispositional trust in doing an action. A dual definition has been proposed to describe *dispositional trust in inaction*, i.e., a situation where the target is trusted not to do an action. Agent i is disposed to trust agent j *not* to do α w.r.t. φ in the circumstances κ iff

- i has the potential goal φ in the circumstances κ
- i believes that from now on, if it has the goal φ and κ holds, then
  - j will be capable to do α
  - j, by doing α, will ensure φ to be *always false*
  - j *does not* intend to do α

The concept of reputation has been addressed as a group belief: reputation refers to a dispositional trust belief of a group of agents instead of a single one. Agent j has reputation in group I to do α w.r.t. φ in the circumstances κ iff

- I has the potential group goal φ in the circumstances κ
- it is public for the group I that from now on, if the group I has the goal φ and κ holds, then
  - j will be capable to do α
  - j, by doing α, will ensure φ to be true at some point
  - j will intend to do α

Because of space constraints, we point the interested reader to [5] for the formalization of such concepts as well as the definition of potential (group) goal and public belief.

B. Reputation not to do an action

Despite defining dispositional trust in a target *doing* and *not doing* an action, Herzig et al. only define reputation of a target doing an action. However, some scenarios require to represent the fact that a target is trusted within a group not to act in a way that would defeat a goal of the group. For this reason, we suggest to extend [5]'s framework with the concept of reputation of an agent not to do an action. More specifically, we propose that agent j has reputation in group I *not* to do α w.r.t. φ in the circumstances κ iff

- I has the potential group goal φ in the circumstances κ
- it is public for the group I that from now on, if the group I has the goal φ and κ holds, then
  - j will be capable to do α
  - j, by doing α, will ensure φ to be *always false*
  - j *does not* intend to do α

For completeness, we formalize the concept of reputation not to do an action according to the formalism introduced in [5].

$$
\begin{aligned}
&Reput(I, j, \sim \alpha, \varphi, \kappa) \stackrel{def}{=} \\
&\quad PotGoal_I^{\forall}(\varphi, \kappa) \wedge Public_I G^*(\\
&\qquad (\kappa \wedge \textstyle\bigwedge_{i \in I} Goal_i(\varphi) \wedge Capable_j(\alpha) \wedge After_{j:\alpha} G^* \neg\varphi) \\
&\qquad \Rightarrow \neg Intends_j(\alpha))
\end{aligned}
$$

IV. Instantiation

In order to apply the conceptual framework described in Section III to the scenario introduced in Section II we have to

- identify the target agent j, the evaluating agent i or group of agents I, the action α of j, the goal φ of i or I and the context κ
- find a way to estimate i's dispositional trust in j (resp. j's reputation in I) (not) to do α w.r.t. φ in the circumstances κ

We will discuss the two issues in Sections IV-A and IV-B respectively.

A. Basic entities

It might be intuitive considering the target agent j as the Wikipedia contributor and the evaluating agent i as the patroller. After thinking a bit, it might be reasonable considering φ as the goal of preserving the integrity of Wikipedia articles which is independent from the circumstances κ. More debatable is which group(s) of agents I and which action(s) α we might want to consider.

W.r.t. α, Krupa et al. [8] consider two possible actions: *good* and *bad edits* (which they call *modification* and *vandalism* respectively). This approach is consistent with Wikipedia's philosophy ("A bad edit is an edit that for one reason or another may need to be entirely removed. A needy edit requires maintenance or improvement in some manner" [12]) and allows them to identify both bad and needy edits, as shown in Tab. I. Tab. I is supposed to be read as follows: rows (resp. columns) represent whether the contributor is trusted to provide a good edit (resp. to refrain from providing a bad edit). Notice that if an edit is trusted to be bad then it does not make sense checking whether it is trusted to be good.

$\sim$ bad edit / **good edit**	**false**	**true**
false	bad	needy
true	bad	good

Table I
[8]'s trust model

W.r.t. I, a trivial solution would be to consider as group of agents the set of all patrollers. However, it might be the case that most patrollers do not have any opinion about a given contributor, since it is unlikely that many patrollers checked many edits of the same contributor. Therefore, for each edit it might be meaningful considering I as the set of patrollers having a sufficiently well-founded opinion of the contributor who did such edit. Although this is the approach enforced in our implementation, we briefly mention a concurrent proposal (somehow suggested by [8]) which leverages articles' categories, i.e., "groups of articles on related topics".[3] According to this approach, when evaluating edits of articles belonging to a given category, I would be the set of patrollers who are expert for that category. We list below some issues which make the implementation of such approach a complex task.

- Not all articles are assigned to some category
- Some articles are assigned to more than one single category
- Category assignment varies with time
- Retrieving patrollers' expertise is not trivial

B. Trust/Reputation estimate

As described in Section III, dispositional trust (resp. reputation) of a target agent j w.r.t. an action α and a goal φ can be computed based on the beliefs of an evaluating agent i (resp. a group of agents I) about: (i) j's capability and intention to do α; and (ii) which consequences j's execution of α has w.r.t. φ. W.r.t. the instantiation presented in Section IV-A, we hence have to estimate whether a given patroller believes (resp. whether it is public in the set of patrollers having a sufficiently well-founded opinion of the contributor) that

1) a given Wikipedia contributor is capable to do a good edit
2) by doing a good edit, the contributor will ensure the article to be consistent at some point

[3] http://en.wikipedia.org/wiki/Wikipedia:FAQ/Categories

3) the contributor intends to do a good edit
4) the contributor is capable to do a bad edit
5) by doing a bad edit, the contributor will ensure the article to be always inconsistent
6) the contributor does not intend to do a bad edit

By definition, a good edit preserves the consistency of an article. Moreover, every contributor is capable to do a bad edit. Finally, although a bad edit leaves an article in an inconsistent state, it is unlikely that the article will stay inconsistent forever. For these reasons, we can assume that every patroller believes (resp. that it is public in the set of patrollers having a sufficiently well-founded opinion of the contributor) that, for each Wikipedia contributor, items 2. and 4. in the list above are true and item 5. is false.[4] Therefore, we only need to find a way to estimate whether the patroller believes (resp. whether it is public in the set of patrollers having a sufficiently well-founded opinion of the contributor) that: (i) the contributor is capable and intends to do a good edit; and (ii) she does not intend to do a bad edit.

Statistics about the previous activity of Wikipedia contributors can be adopted as a starting point. For instance, w.r.t. point (ii), since every Wikipedia contributor is capable to do a bad edit, whenever a given Wikipedia contributor j does a bad edit it holds that she *intended* to do a bad edit. For this reason, j can be assumed not to intend to do a bad edit whenever the number of bad edits she did is higher than a given (absolute or relative) threshold. Similarly, j can be assumed to be capable and to intend to do a good edit whenever the number of good edits she did is higher than a given threshold (not necessarily equal to the previous one).

This approach requires that, for each Wikipedia contributor j, the number of her (good/bad) edits is retrieved. The overall number of edits (starting from a given timestamp) can be easily retrieved by monitoring the edits occurring on Wikipedia and counting the ones done by j. More complex is to retrieve the number of good/bad edits, since patrollers do not explicitly tag edits as good, needy or bad. Krupa et al. rely on heuristics: they consider an edit to be good (resp. bad) if the time elapsed between j's edit and the following one related to the same article is above a given threshold (resp. the same article has been modified right after by some patroller who left a comment containing either of the words `rv`, `revocation`, `révocation`, `vandalisme` and `annulation`—notice that they focused on the French Wikipedia). Moreover, an article's category is taken into account when evaluating whether an edit done on such an article should be considered good.

Unfortunately, Krupa et al. do not provide evidences showing that their heuristics are not only reasonable but also effective. Indeed, beside the problems related to Wikipedia categories we mentioned at the end of Section IV-A, such heuristics could in principle be affected by the following issues: it might be the case that a given article is modified right after a good edit (resp. that a bad edit is discovered only after a while), moreover it might be the case that a patroller did not provide any comment when reverting a bad edit or that the comment does not contain the expected keywords. For this reason, at least two directions are feasible as further work: (i) performing a quantitative analysis in order to evaluate the effectiveness of such heuristics; and (ii) enabling patrollers to provide explicit feedback. Being the quantitative analysis still ongoing work (cf. Section VIII), Section V describes an implementation of the second approach based on the Jason [1] and CArtAgO [13], [14] frameworks.

V. Implementation

In order to simplify the activity of patrollers, a number of tools and resources are available (a list thereof is provided in [12]). Some of them filter the edits on Wikipedia articles according to some criterion (e.g., by focusing on the activity of contributors inside/outside the patroller's watch list), whereas most of them provide patrollers with all edits performed in some subset of Wikipedia (e.g., en.wikipedia). Some of them use as input Wikipedia's "Recent changes" special pages or RSS feed, but the favourite source of information about latest edits are by far Wikipedia's "Recent changes" IRC channels hosted on irc.wikimedia.org which communicate in real-time every operation performed on Wikipedia.

Tools currently available have been mainly designed to support the activity of patrollers who work independently from each other and do not allow for knowledge transfer among them. This section introduces Mousquetaire[5], a collaborative tool enabling knowledge transfer among patrollers. Such a tool implements the approach described in Section IV and is freely available at http://labh-curien.univ-st-etienne.fr/~decoi/Mousquetaire.zip. Section V-A presents Mousquetaire's user interface, whereas Section V-B provides information about its implementation.

A. User interface

Mousquetaire's GUI (depicted in Fig. 1) has been inspired by the one of Vandal Fighter[6]. Both of them: (i) show edit notifications posted on Wikipedia's "Recent changes" IRC channels as rows in a table; (ii) allow to click on such rows in order to open the system browser at a page comparing the versions of the article immediately preceding and following the corresponding edit; and (iii) dynamically

[4]Notice that this is different than in [8], where items 1. and 3. (but not items 2. and 4.) are assumed to be true.

[5]The name has been suggested by the musketeers' collaboration-oriented motto *Un pour tous! Tous pour un!* ("One for all, all for one") in Alexandre Dumas' novel *Les Trois Mousquetaires* ("The Three Musketeers").

[6]http://en.wikipedia.org/wiki/Wikipedia:VF

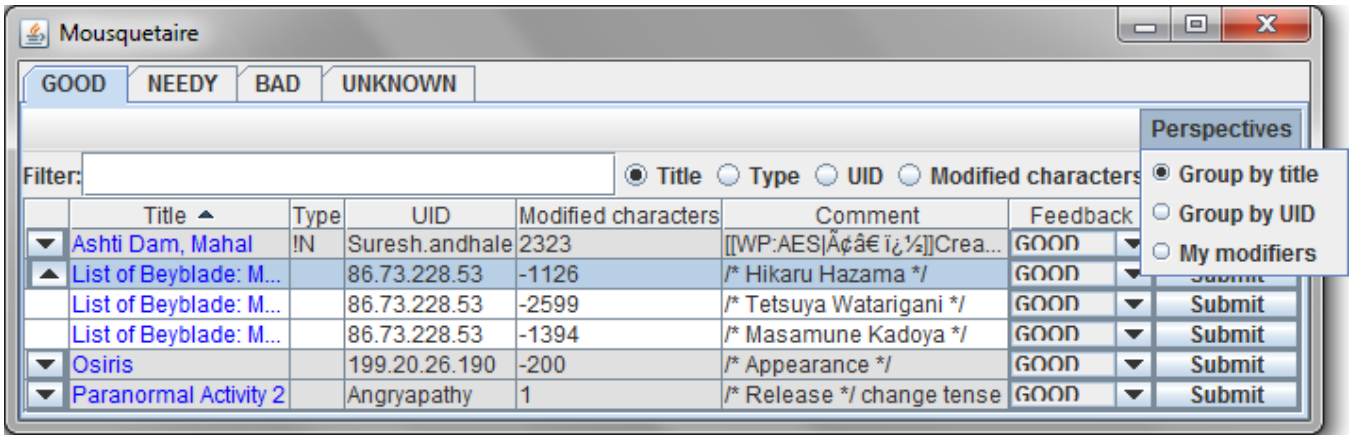

Figure 1. MOUSQUETAIRE's GUI

fill the table with new rows as soon as edits are notified through Wikipedia's IRC channels.

MOUSQUETAIRE's GUI also provides further functionalities: (i) table rows can be ascending/descending ordered according to the values appearing in a given column; (ii) rows can be filtered in order to show only the ones containing a given string in a given column; (iii) different perspectives are available which allow to group rows by article or contributor so that only the most recent edit per group is shown. In case such a perspective is selected, older edits of a given article (resp. by a given contributor) can be shown/hidden by clicking on the arrow at the left of the group's most recent edit.

Yet another perspective allows to only show further edits of articles edited by the patroller. This feature is especially useful to identify: (i) contributors who keep vandalizing articles even upon subsequent reverts of their edits; (ii) edit wars, which occur "when editors who disagree about the content of a page repeatedly override each other's contributions, rather than trying to resolve the disagreement by discussion".[7]

However, the by far most important difference between MOUSQUETAIRE's GUI and Vandal Fighter's one is that the former allows patrollers to provide feedback for a given edit by selecting either of `GOOD`, `NEEDY` or `BAD` in the corresponding combo box and pressing the corresponding `Submit` button. Patroller feedback is then leveraged in order to automatically classify upcoming edits and show them in the corresponding tab.

B. Behind the scenes

Fig. 2 shows MOUSQUETAIRE's main components as well as the actors it interacts with: the front-end to the IRC channel and the IRC channel itself, the GUI, the knowledge bases containing trust/reputation estimates, the patroller and

[7]http://en.wikipedia.org/wiki/Wikipedia:Edit_warring

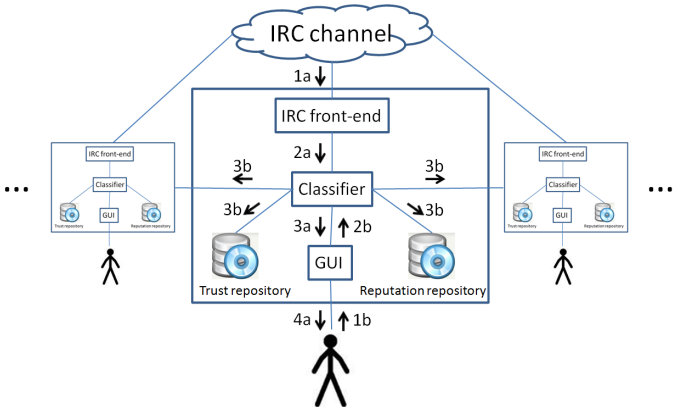

Figure 2. Control flow within a MOUSQUETAIRE engine

the *classifier*, a component in charge of classifying incoming edits according to trust/reputation estimates of the contributors who did the edit. Fig. 2 also shows the interplay among such entities as triggered by the two main events MOUSQUETAIRE deals with: a new edit notification posted on the IRC channel and patroller feedback.

In the first case (identified by letter `a`), the new edit intercepted by the IRC front-end (1a) and provided to the classifier (2a) is assigned by the latter an estimate based on the contributor's trust and/or reputation as available in the corresponding repositories. Both edit and estimate are then forwarded to the GUI (3a) with a request to update the table in the corresponding tab. If the tab is currently selected, the updated table is shown to the user (4a).

In the second case (identified by letter `b`), patroller feedback is communicated through the GUI (1b) to the classifier (2b), which will then use it in order to compute an updated trust/reputation estimate of the contributor whose edit has just been checked by the patroller. Such estimate is eventually stored into the corresponding repository and propagated to peer MOUSQUETAIRE engines (3b).

Section V-B2 (resp. Section V-B3) is devoted to the description of the implementation of the control flow identified by letter `a` (resp. `b`). Since the classifier is implemented as a Jason agent, whereas both the IRC channel and the GUI are wrapped by a CArtAgO artifact, Section V-B1 briefly recalls such technologies.

1) Back-end technologies: The code of a Jason [1] agent specifies its: (i) beliefs; (ii) proactive behavior (i.e., the tasks which the agent will perform *sponte sua*); and (iii) reactive behavior (i.e., the tasks which the agent will perform as a reaction to some event occurred in the outer world). Agent beliefs are represented as a Logic Programming-like [9] knowledge base. Agent behavior is specified as a set of plans, each of which consists of a list of activities to be performed if some condition holds. Examples of such activities are to query or modify the agent's knowledge base and to interact with other agents or the environment.

The CArtAgO [13], [14] framework greatly simplifies the interaction between Jason agents and the environment. In particular, CArtAgO allows to populate the environment with artifacts whose functionalities can be invoked by Jason agents and which can themselves issue signals, possibly triggering the reactive behavior of Jason agents. CArtAgO artifacts are Java classes extending `cartago.Artifact`: they can issue signals by invoking the parent class's `signal` methods, whereas Jason agents can directly invoke methods of CArtAgO artifacts annotated with `cartago.OPERATION`.

2) New edit operation: The control flow identified by letter `a` in Fig. 2 has been implemented as follows: whenever the CArtAgO artifact wrapping the IRC channel is notified that a new edit occurred, it issues the signal `handleNewEntry` with two parameters: an object containing information about the occurred edit and the userId (or IP address) of the contributor. The Jason agent implementing the classifier reacts to such a signal as follows: first the agent's knowledge base is inspected in order to retrieve an estimate of the contributor. Afterwards, the method `addEntry` of the CArtAgO artifact wrapping the GUI is invoked with two parameters: the estimate and the object containing information about the occurred edit.

In MOUSQUETAIRE, estimates of Wikipedia contributors can be based either on dispositional trust or reputation. In the first case, a patroller believes a contributor to be `unknown` if the patroller has no information about her. Otherwise, the patroller believes the contributor to be: (i) `good` if the number of good edits she did is higher than the half of her overall number of edits; (ii) `bad` if the number of bad edits she did is higher than the half of her overall number of edits; (iii) `needy` otherwise.

In the second case, the reputation of a contributor is: (i) `good` if all patrollers who already evaluated some edit of hers believe her to be `good`; (ii) `bad` if all such patrollers believe her to be `bad`; (iii) `needy` otherwise. Notice that if a patroller never evaluated any edit of a contributor she does not belong to the group within which the contributor has a given reputation, therefore she will assume such a contributor to have an `unknown` reputation.

3) Patroller feedback: The control flow identified by letter `b` in Fig. 2 has been implemented as follows: whenever the CArtAgO artifact wrapping the GUI is notified that the patroller submitted her feedback about some edit, it issues the signal `feedback` with two parameters: the userId (or IP address) of the contributor and the feedback submitted by the patroller. Within the knowledge base of the Jason agent implementing the classifier, a `history/4` fact is available for each evaluated contributor. Beside the contributor's userId (or IP address), such a fact contains the number of her edits which have been evaluated as good/bad along with her overall number of edits.

When reacting to a `feedback` signal, the agent first checks whether information about the contributor who did the evaluated edit is available in its knowledge base. If this is the case, the `history/4` fact related to the contributor is updated in both the agent's knowledge base and the other agents' one. The second step requires to broadcast two messages, the first one retracting the outdated information and the second one asserting the updated one.

If information about the contributor who did the evaluated edit is not available in the agent's knowledge base, a new `history/4` fact is added to both the agent's knowledge base and the other agents' one.

VI. EXPERIMENTAL EVALUATION

As described in Section V, MOUSQUETAIRE can work both in the dispositional trust and reputation modalities. This section describes an experimental evaluation of the dispositional trust modality, whereas an evaluation of the reputation one is regarded as further work.

The results provided in the following are based on data crawled from the IRC channel devoted to the English Wikipedia throughout 24 hours starting from Fri Oct 22 16:45:49 CEST 2010. The dataset contains information about 183386 edits performed on 103811 Wikipedia pages by 30806 contributors. In average, 2.12 edits per second occurred, each page was edited 1.77 times and each contributor edited 5.95 times. Most pages were edited no more than once and most contributors edited no more than once. On the other hand, most edits involved 19576 pages (i.e., 19% overall) and were performed by 660 contributors (i.e., 2% overall).

103146 edits (i.e., 56% overall) involved Wikipedia pages other than articles (namely, books, categories and other administrative pages, files, portals, templates and users as well as talks about each of them). 28528 edits (i.e., 16% overall) were performed by bots, i.e., "automated or semi-automated tools that carry out repetitive and mundane tasks in order to

maintain the [...] articles of the English Wikipedia".[8]

The statistics provided in the following only relate to edits performed by humans on articles. 3825 edits have been manually reviewed in order to assess their quality. Assessing the quality of an edit is not always trivial, since knowledge about the edited article's topic is required. For this reason, in order to provide for an assessment as uniform as possible we mainly focused on syntactical issues. As a consequence, we considered edits contradicting each other as *good* as long as they left the article in a grammatically consistent state.

As a result of the assessment, each edit has been assigned a label out of `G`, `N`, `P`, `B` or `U`. Labels `G`, `N` and `B` are supposed to identify *good*, *needy* and *bad* edits respectively. Label `P` (*possibly bad*) has been assigned to suspicious edits which did not seem to deserve the label `B`. Finally, label `U` (*unknown*) has been assigned to edits which could not be checked, since an attempt to retrieve them returned an error message from the Wikipedia database.

Our evaluation only considered edits of contributors who performed at least two edits not labeled as *unknown* (2944 edits by 398 contributors). Fig. 3(a) (resp. Fig. 3(b)) provides a report about the effectiveness of our approach where *possibly bad* edits are considered as *good* (resp. *bad*). Beside noticing that the two reports do not essentially differ, we observe that our approach has a remarkable precision (86-87% of estimates are correct), whereas 5% of the estimates are false negatives (i.e., *good* edits considered as *bad*) and 4-5% are false positives (i.e., *bad* edits considered as *good*).

We conclude this section by mentioning that the MOUSQUETAIRE package described in Section V also contains: (i) a database storing the crawled Wikipedia edits as well as utilities to automatically create such a database; (ii) a database storing the filtered Wikipedia edits as well as utilities to automatically filter out edits performed by bots or on Wikipedia pages other than articles; (iii) a database storing the manually assessed edits as well as utilities enabling to reproduce the results shown in Fig. 3; and (iv) a `readMe.txt` file explaining how to use the above-mentioned utilities.

VII. Related work

The freedom to contribute that governs Wikipedia is a major reason of its success. However, it also enables to damage the encyclopedia by vandalizing its articles. In such a case, reverting bad edits requires first of all to identify them. Some tools, such as VandalFighter, notify patrollers of new edits. However, because of Wikipedia's intense activity, they cannot check every modification but have to define their own heuristics to decide which one should (not) be checked.

Automatic assistance is hence desirable to identify bad or needy edits as accurately as possible. A competition on vandalism detection software has been organized and analyzed in [11]. Most of the competing tools classify edits according to features like presence of given keywords, size of the modifications, capitalization of characters or topic of the article. However, only one [6] of the nine participants to the competition exploits an estimate of the contributor's trustworthiness. This approach has shown promising results, despite the trust model being a simple one.

Indeed, the decision (not) to check an edit is essentially based on trust. However, such decision is subjective as different patrollers might take different decisions for the same edit. Trust management has been addressed in the field of multi-agent systems and some work has been proposed which takes the social aspect of trust into account. A survey of existing models in this very active field can be found in [15].

Most of the trust models for multi-agent systems rely on statistics to make automatic decisions. However, this approach shows its limits whenever a tight interaction with human users is required as the integration of human and software trust decision processes has proved to be difficult. To overcome this drawback, socio-cognitive models of trust and reputation have been proposed [3], [4]. The work described in this paper builds this late approach into a software agent assisting patrollers in their subjective trust decision and enabling them to collaboratively build reputations.

VIII. Conclusions and further work

The wide diffusion of open and decentralized environments like the Web makes it possible for actors to interact with previously unknown peers. As a consequence, trust has become a hot topic in the field of computer science. Many attempts to formalize concepts like *trust* and *reputation* have been carried out in the literature, most remarkably the one by Herzig et al. [5].

However, Herzig et al. focus on describing a conceptual framework but do not provide any concrete instantiation of it, thereby not showing any evidence about the effectiveness of their approach. In this paper, we filled the gap by: (i) extending the conceptual framework described in [5]; (ii) presenting an instantiation of such conceptual framework; and (iii) describing an implementation of such instantiation based on agent technologies. Although our implementation targets a Wikipedia-related scenario and exploits the Jason [1] and CArtAgO [13], [14] frameworks, the methodology we presented is general and can be applied to different scenarios and agent technologies.

Concerning current and further work, the first issue we are trying to address is the cold-start problem: new edits can be automatically classified as good, needy or bad only if information about their contributors is available to the patroller('s group), i.e., if the patroller or someone in her group already evaluated some edit of the contributors. Heuristics like the ones presented in [8] provide a possible way to overcome this problem. For this reason, we are currently assessing the

[8] http://en.wikipedia.org/wiki/Wikipedia:Bots

actual / expected	bad	needy	good
bad	9%	0%	4%
needy	2%	0%	2%
good	5%	0%	77%

(a) *possibly bad* edits are considered as *good*

actual / expected	bad	needy	good
bad	11%	0%	5%
needy	2%	0%	2%
good	5%	0%	76%

(b) *possibly bad* edits are considered as *bad*

Figure 3. Results of the evaluation of MOUSQUETAIRE's dispositional trust modality

quality of edits in a manual way in order to collect statistics to be used for the definition of meaningful heuristics.

As a second goal, we would like to make our implementation available to the Wikipedia community. If on the one hand real users would provide the ideal testing environment for our approach, on the other hand our implementation needs to undergo some technical improvements before being publicly released. First of all, the user-friendliness of the GUI should be improved: so far, patrollers are required to provide explicit feedback about the edits they check. A future version could inspect a patroller's edit and automatically consider the previous one related to the same article as: (i) good; (ii) needy; or (iii) bad; according to whether the patroller: (i) checked the edit and did not do anything else; (ii) improved the edit; or (iii) reverted it.

ACKNOWLEDGMENTS

The work described in this paper has been carried out within the scope of the ANR ForTrust[9] project ANR-06-SETI-006.

REFERENCES

[1] Rafael H. Bordini, Jomi Fred Hübner, and Michael J. Wooldridge. *Programming Multi-Agent Systems in AgentSpeak using Jason*. John Wiley & Sons, 2007.

[2] Michael E. Bratman. *Intention, Plans, and Practical Reason*. Cambridge University Press, March 1999.

[3] Cristiano Castelfranchi, Rino Falcone, and Emiliano Lorini. A non reductionist approach to trust. In J. Golbeck, editor, *Computing with Social Trust and Reputation*, Human-Computer Interaction Series, pages 45–72. Springer, 2008.

[4] Rosaria Conte and Mario Paolucci. *Reputation in Artificial Societies. Social Beliefs for Social Order*. Boston: Kluwer, 2002.

[5] Andreas Herzig, Emiliano Lorini, Jomi Fred Hübner, and Laurent Vercouter. A logic of trust and reputation. *Logic Journal of the IGPL*, 18(1):214–244, 2010.

[6] Sarah Javanmardi. Submission to the 1st international competition on wikipedia vandalism detection, 2010. from the University of California, USA.

[9] http://www.irit.fr/ForTrust/

[7] Andrew J. I. Jones. On the concept of trust. *Decis. Support Syst.*, 33(3):225–232, 2002.

[8] Yann Krupa, Laurent Vercouter, Jomi Fred Hübner, and Andreas Herzig. Trust based evaluation of wikipedia's contributors. In *ESAW*, pages 148–161, 2009.

[9] John W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.

[10] Stephen Marsh and Pamela Briggs. *Computing with social trust*, chapter Examining trust, forgiveness and regret as computational concepts. Springer, 2009.

[11] Martin Potthast, Benno Stein, and Teresa Holfeld. In Martin Braschler and Donna Harman, editors, *Notebook Papers of CLEF 2010 LABs and Workshops, 22-23 September, Padua, Italy*.

[12] Recent changes patrol. http://en.wikipedia.org/wiki/Wikipedia:Recent_changes_patrol.

[13] Alessandro Ricci, Michele Piunti, Mirko Viroli, and Andrea Omicini. Environment programming in CArtAgO. In *Multi-Agent Programming*, pages 259–288. 2009.

[14] Alessandro Ricci, Mirko Viroli, and Andrea Omicini. CArtAgO: A framework for prototyping artifact-based environments in MAS. In *E4MAS*, pages 67–86, 2006.

[15] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artif. Intell. Rev.*, 24(1):33–60, 2005.