Improving the efficiency of collaborative work with trust management

Xingyu Zheng, Patrick Maillé, Cam Tu Phan Le, Stéphane Morucci

Improving the Efficiency of Collaborative Work with Trust Management

Xingyu Zheng, Patrick Maillé
Institut Telecom; Telecom Bretagne
2, rue de la Châtaigneraie CS 17607
35576 Cesson Sévigné Cedex, France
Email: {xingyu.zheng},{patrick.maille}@telecom-bretagne.eu

Cam Tu Phan Le, Stéphane Morucci
Swid
80, avenue des Buttes de Coesmes
35700 Rennes, France
Email: {stephane.morucci},{plcamtu}@swid.fr

***Abstract*—The concept of trust has recently been introduced in the context of peer-to-peer networks, in order to deal with uncertainty regarding the behavior of imperfectly known agents. In this paper, we apply the notion of trust to situations of collaborative work, more precisely of document editing.**

We suggest to use trust to compute a satisfaction score for each participant, in order to help select the successive editors of the document so as to avoid unnecessary readings by all collaborators after each modification. We assume that the document development process ends when all collaborators are satisfied with the quality of the document. Two mechanisms using trust to improve that process are proposed, and compared to the situation without trust. Extensive simulations suggest that trust can improve the efficiency of the collaborative work, while it can be implemented in a distributed manner.

I. Introduction

Many types of documents cannot be entirely produced by a single individual, and instead need to be jointly built by several collaborating participants. One can think of examples from diverse working sectors: law texts, research articles, project proposals, course textbooks, or encyclopedias (the most famous example being the online encyclopedia Wikipedia [www.wikipedia.org]). In all those cases, collaborative working is needed because of the prohibitive size of the document to be produced, and/or because of the diversity of the knowledge fields and competences involved.

There exists a variety of tools to facilitate collaborative work, such as online wikis or versioning softwares. Those are not indifferent to the outcomes of the collaborative working activity: it has for example been shown experimentally that structuring the processes in collaborative work systems improves the team results as well as the participant satisfaction [1]. Likewise, some advanced tools can be implemented to improve group awareness, parallelism, group memory, and coordination among collaborative workers [2].

Now, an important challenge for collaborative work is to determine when and how to stop the document development process. Do all participants need to approve the final version? If so, when can it be considered that a version is a candidate for being sent for approval? Such questions might be difficult to answer when the number of collaborators becomes large. Likewise, it is not reasonable to assume that all collaborators go through the whole document after each modification by one of her peers, therefore some more fine-tuned management methods could significantly improve the development process efficiency.

In this paper, we suggest to address those questions by using the notion of trust [3]. Several kinds of trust usages have been proposed, mainly for grid computing environments and peer-to-peer systems. Indeed, in those cases where each node shares its resources and benefits from those offered by its peers, trust mechanisms can be used to find the best peers to exchange with, or to avoid malicious nodes [4], [5], [6], [7]. Remark moreover that inter-company (or even inter-personal) collaborations often rely on the (most often implicit) notion of trust, that can also be modeled and somehow automated in order to make the right business decisions [8].

We propose here a new trust-based mechanism aimed at optimizing the process of collaborative document building. Our objective is to use trust as a tool to limit the total time and the total effort spent to reach the common objective of the group. More precisely, we intend to avoid unnecessary readings, and efficiently select editors during the development process.

The remainder of this paper is organized as follows. In Section II, we introduce the basic concepts of trust and show how it can be used to compute user satisfaction scores in a collaborative working environment model. Section III presents three models of collaborative working processes, where trust can be used to select the successive editors of the document. The performance of those schemes are compared through simulations in Section IV. We discuss the technical feasibility of our mechanism in Section V, and conclude and provide directions for future work in Section VI.

The results presented here follow the same model as the one we introduced in [9]. However, here we provide more complete performance results, considering a richer set of parameter values and also looking at a fairness measure (the effort repartition among participants). Moreover, with respect to the work of [9], this extended version describes an effective implementation of the proposed mechanisms through an OpenOffice plugin.

II. Using trust in a collaborative work environment

We consider a working system with a number $I = |\mathcal{I}|$ of agents who collaboratively contribute to the elaboration of a document. Those agents successively improve the document in an asynchronous manner, until it reaches a satisfying quality: an agent elaborates a first version of the document, that is afterwards edited and modified by the other agents in a certain order.

A. Definition of trust

Trust can be defined as a quantification of the confidence that an agent has in another agent to behave in an appropriate manner. Thus, trust represents the beliefs of one agent in the future actions of the other. In most trust-based models [10], [11], [12], [13], that quantification is summarized into a number in the interval $[0, 1]$. Consequently, trust values may also be interpreted as probabilities of the trusted agent to behave properly, from the point of view of the truster agent. We denote by $\mathcal{I}$ the set of agents, and for $i, j \in \mathcal{I}$, by $t_{i,j}$ the trust value representing how much agent i trusts agent j.

The following conventions are classically taken:

- a trust value $t_{i,j}$ of 1 means that agent i perfectly trusts agent j, while $t_{i,j} = 0$ means that agent i does not trust j at all;
- the trust value $t_{j,i}$ can be different from $t_{i,j}$, as trust scores materialize some subjective (thus, possibly biased) perceptions;
- $t_{i,i}$, representing the self-trust, is fixed to 1.

For an agent $i \in \mathcal{I}$, there are two ways of estimating how much she trusts another agent $j \in \mathcal{I}$.

- Agent i can build her own trust score $t_{i,j}$ based on her experience from previous interactions with j. In peer-to-peer systems, agent i can for example keep a record of satisfying and unsatisfying exchanges with peer j, possibly in the form of a weighted sum, where weights can be used to take into account the importance of the interaction or the time aspect [14]. In collaborative work, one can think of an evaluation score based on the contributions made by the collaborator in previous common works, which can also be quantified [15], [16].
- If agent i does not have a sufficiently large history of interactions with j to compute an accurate trust estimation, she can ask some of her trusted agents to provide her with a *recommendation score*, and then combine the received values with her own trust in those recommenders to calculate $t_{i,j}$ [17].

In this paper, we do not consider that aspect of trust estimation; we assume that trust scores $(t_{i,j})_{(i,j)\in\mathcal{I}^2}$ have been determined, and we rather focus on the use of those scores in collaborative work environments so as to optimize the joint work process.

B. Agent satisfaction scores during the collaborative work

An important feature of our model is that each editor $k \in \mathcal{I}$ has to associate to the document an evaluation score e_k after her editing the document. That score, that we impose to be in the interval $[0, 1]$, represents the subjective quality of the document for editor k. We assume here collaborators to be honest, so that they truthfully reveal the value of the document quality. Taking into account other reporting strategies driven by maliciousness or selfishness is above the scope of this paper.

The evaluation score is meant to be used by all the other collaborators, who combine it with their trust in the last editor of the document to compute a personal *satisfaction score* for the document. Formally, if $\mathrm{Id}_{\text{last}}$ is the identity of the last document editor, each agent $i \in \mathcal{I}$ automatically computes the satisfaction score

$$s_i := t_{i,\mathrm{Id}_{\text{last}}} \times e_{\mathrm{Id}_{\text{last}}}, \tag{1}$$

without even opening the document.

This means that the participants take the declared score $e_{\mathrm{Id}_{\text{last}}}$ into account, but are less optimistic in following the opinion of $\mathrm{Id}_{\text{last}}$ when they do not trust that agent. In that sense, the multiplicative combination rule can be interpreted as providing a lower bound of the quality of the document: the document can be of high quality, but before agent i verifies that through a careful reading, she remains cautious and lowers the declared evaluation score, at a greater scale if she does not trust the last editor. Intuitively, this consists in viewing $t_{i,\mathrm{Id}_{\text{last}}}$ as the trust value of i in $\mathrm{Id}_{\text{last}}$ correctly estimating the quality of the document. An incorrect (from i's point of view) evaluation may be due to the last editor not being aware of all the elements in the document, or to agents i and $\mathrm{Id}_{\text{last}}$ having different priorities and interests. That last possibility might be problematic in our case, since there would be here an incentive for editors to overestimate the evaluation. Therefore, in this paper we assume that the final objective of all agents is the same (namely, create a high quality -with the same standards- document), and that evaluation discrepancies do not stem from malicious behaviors but rather from imperfect perceptions.

III. Management of collaborative working through trust

In this section, we introduce three different ways of managing the collaborative working environment. By management, we mean here the process of selecting the successive document editors.

Note that we implicitly assume that agents never edit the document at the same time, while most current collaborative work systems offer that possibility. However, in those cases the last agent to submit her version to the system is asked to deal with potential inconsistencies with the changes that have occurred since she last downloaded the document. That checking procedure could then be interpreted as an extra reading of the document, which would somehow be similar to the situation where agents successively modify the document.

A. Collaborative work model

We describe here the condition to end the editing process. We assume that each agent $i \in \mathcal{I}$ has a threshold $b_i \in [0, 1)$

that she considers to be the minimal satisfaction level for the document to be acceptable. Following the idea of collaborative work as a way to reach a consensus among participating agents, we consider that the document editing process stops when all agents are satisfied with the quality of the document, i.e., when the satisfaction scores $(s_i)_{i\in\mathcal{I}}$ are such that

$$s_i \geq b_i \qquad \forall i \in \mathcal{I}. \tag{2}$$

Until that condition is satisfied, we assume that users go on successively editing the document.

We immediately notice that the process as we defined it may never converge: assume two agents $i \neq j$ are such that their trust values for the other one are below their threshold level, i.e.,

$$\forall k \in \mathcal{I}, \quad \begin{cases} k \neq i \Rightarrow t_{i,k} < b_i \\ k \neq j \Rightarrow t_{j,k} < b_j. \end{cases}$$

Then in that case, the condition (2) can never be met since evaluation scores e_k are in the interval $[0, 1]$.

For that reason, we slightly modify the calculation mode of satisfaction values, by introducing an aspect related to the document history. From now on, we consider that satisfaction scores $(s_i)_{i\in\mathcal{I}}$ are computed according to the relation

$$s_i := \min\Big[(t_{i,\mathrm{Id}_{\mathrm{last}}} + \alpha n_{\mathrm{edit}}(i)) \times e_{\mathrm{Id}_{\mathrm{last}}} \ , \ 1\Big], \tag{3}$$

where

- $n_{\mathrm{edit}}(i)$ is the number of edits by agent i in the process, and
- α is the bias in the satisfaction calculation, that corresponds to each extra edit of the document.

The rationale behind that new expression is as follows: Since we assumed that agent evaluations are aimed at a common objective, and only differ because of imperfect knowledge of the document and of quality criteria, we can reasonably assume that the more an agent has gone through the document, the more confident she is in its quality. We upper bound the result by 1, so that satisfaction scores remain in the interval $[0, 1]$.

Note that the new expression (3) ensures that the document development process stops if all unsatisfied agents have the possibility to edit the document.

In the remainder of this section, we describe the three development process management algorithms that are studied in the rest of the paper.

B. A naïve scheme: pure round-robin document editing

The first mechanism considered does not actually use trust values to manage the document development process. We only use satisfaction scores to decide when to stop the process, but the order in which participants edit the document is fixed. More precisely, we assume in that case that the "edit token" circulates in a round-robin fashion: if agents are numbered $\{1, 2, ..., I\}$ (up to a permutation), then the identity of the editor just follows a simple rotation pattern $1, 2, ..., I, 1, 2, ...$, until condition (2) is met.

Later in this paper, we will also assume that agent i may not always be available when it is her turn to edit the document. In the naïve scheme described here, we assume that in that case the token immediately goes to the next agent (i.e., the agent $[(i \mod I) + 1]$), so that no time is lost waiting for i to be available again.

C. Using trust to choose the next editor: round-robin among unsatisfied agents

In this subsection, we propose to adapt the peer-to-peer strategy consisting in selecting, as the peer to download files from, one peer among all sufficiently trusted peers (i.e., those for which the trust level is above a given threshold).

In our collaborative work context, this can be translated into the following: the next editor of the document is still chosen according to a round-robin scheme, but only among agents i for which the current satisfaction value is below their quality threshold b_i. In other words, we simply follow the scheme of the previous subsection, but skipping participants who are already satisfied with the current document quality.

Interestingly, that scheme can be completely decentralized. The circulation of the edit token can follow a pre-specified pattern as described in the previous subsection, with satisfied participants immediately transmitting it to their successor without modifying the document. (Notice that each participant computes her satisfaction score locally, based on the declared evaluation score and her trust values in the collaborators.) Moreover, detecting the end of the process is made easier: actually no agent will edit the document anymore if condition (2) is satisfied, so there is in any case no risk of unnecessary edits. To officially stop the process, one can imagine that if a participant receives the document and notices she was the last editor, then this means all other participants were satisfied with the document, and thus she can declare the process as terminated. In the case when collaborators are not always available, a more developed method needs to be defined. A possibility would be to automatically add to the document a flag stating that the agent was unsatisfied with the current version (when asked to edit it) but was unavailable; that method would avoid wrongly stopped processes.

D. Having the least satisfied agent improve the document

We now follow a different strategy inspired by peer-to-peer networks, where peers choose to download their requested files from the host with the highest trust value [4]. We adapt that policy to the context of collaborative work, by giving the edit token to a collaborative agent k for whom the satisfaction score is the furthest below her threshold, i.e., the next editor is the (an) agent k such that $s_k - b_k = \min_{i\in\mathcal{I}}(s_i - b_i)$.

Intuitively, that third scheme should be more efficient than the one defined in Subsection III-C, since we choose to specifically target the minimal *satisfaction minus threshold* value, and the process ending precisely depends on that minimal value (it stops when $\min_{i\in\mathcal{I}} s_i - b_i \geq 0$).

On the other hand, the potential extra efficiency of that scheme will have a price in terms of applicability, since

implementing it in a fully decentralized way becomes difficult. Some possible solutions to choose the next editor according to that policy are given below.

- A central entity can be used to collect all satisfaction scores and notify the next editor. The satisfaction scores could be computed locally by each agent and transmitted to the entity. We can also imagine that they be computed within the central entity to minimize communication exchanges, but that would imply that each participant $i \in \mathcal{I}$ uploads her trust vector $(t_{i,1}, ..., t_{i,I})$ to the entity, which might not be well perceived: since trust values are private and sensitive data, agents may be reluctant to reveal them to the system, even with the assurance that they will not be disclosed to the other collaborators. Moreover, that solution would have the classical drawbacks of centralized systems, in terms of fault tolerance and communication overhead.
- For the implementation to remain decentralized, a full rotation among participants without editing could be performed to collect the identities of the candidates for being the next editor. More precisely, in the document metadata, the identity of the current least satisfied collaborator k and the value of her dissatisfaction $b_k - s_k$ are checked out and updated if necessary, so that at the end of the round the document carries the identity of the next editor. While that scheme is decentralized, it still has the drawback of revealing some information about the trust value $t_{k,\text{Id}_{\text{last}}}$, since all agents know the evaluation score $e_{\text{Id}_{\text{last}}}$ and thus have access to the dissatisfaction level of user k.

Due to those extra difficulties with respect to the mechanism proposed in Subsection III-C, we should be inclined to prefer that scheme over the simpler round-robin among unsatisfied clients, only if the efficiency gain of the development process is significant. Actually, the performance evaluation that we present in the next section does not show such a large gain.

IV. Experiments

This section presents some simulation results aimed at evaluating the performance of the three management methods of Subsections III-B, III-C, and III-D.

A. Simulation model

The model we consider in this paper has several parameters, namely, the set of participants $\mathcal{I}$, the trust values $(t_{i,j})_{i,j\in\mathcal{I}}$, and the bias α that previous edits introduce in the satisfaction score computation (3). We moreover describe now how we model the participant behavior.

1) Trust values and satisfaction score computation: Trust values $(t_{i,j})_{i,j\in\mathcal{I}}$ are chosen randomly, according to a uniform distribution on an interval $[t_{\min}, 1]$. In our simulations, we consider different situations, with $t_{\min} = 0.2$, $t_{\min} = 0.5$, and $t_{\min} = 0.8$ to model low-trust and high-trust communities.

As regards the computation of the satisfaction score s_i after each document modification, we follow relation (3), where the bias α equals 0.1 in our simulations. Similarly, the quality threshold of each participant i is fixed to the same value

$$b_i = 0.9 \quad \forall i \in \mathcal{I}.$$

2) Contributor's availability: In a collaborative working system, the participants may be online or offline over time. To model that aspect, we discretize time into slots (a slot corresponding to an agent editing the document), and consider that at each time slot, each participant is available with a given probability, independently of all other events. Unless otherwise stated, we fixed that availability probability to 0.5 in our simulations. With respect to the problem of contributors not being available, our three management schemes behave as follows:

- **Naïve (round-robin) scheme**: if the editor who is supposed to edit the document is not available, then the token immediately goes to the next one and no time slot is lost.
- **Round-robin among unsatisfied agents**: if the next unsatisfied agent is unavailable, then the token goes to the next unsatisfied agent in the predefined order. If all unsatisfied agents are unavailable, then the time slot is lost (nobody edits the document).
- **Next editor=least satisfied agent**: the edit token immediately goes to the unsatisfied participant i with the minimum value of $s_i - b_i$, *among available participants*. As a result, like for the previous scheme the time slot is lost when all unsatisfied agents are unavailable.

3) Computation of the evaluation score: In a real implementation of our mechanisms, the evaluation score e would be set by the editor after her editing the document, according to her personal competences and knowledge of quality criteria. In our simulation model, we introduce some asymmetry among participants by associating to each one $i \in \mathcal{I}$ a *performance level* $q_i \in [0, 1]$, that can represent the "writing quality" of participant i. That performance level q_i is here randomly chosen for each participant $i \in \mathcal{I}$, according to a uniform law on $[0.5, 1]$ in our simulations.

We then assume that each participant improves the quality of the document -at least from her point of view- when editing it, and that this improvement increases with her writing quality. More precisely, in our simulations the evaluation score e_k set by editor k is computed according to the formula

$$e_k := \; s_{k,\text{prev}} + (1 - s_{k,\text{prev}}) \times q_k, \tag{4}$$

where $s_{k,\text{prev}}$ is the satisfaction value that k had just before editing the document.

B. Performance measures

The efficiency of the document development process is measured by three performance metrics:

- the total duration of the document development process, that is measured by the number of iterations until a satisfying document quality is reached;
- the total effort spent by the collaborating community during the whole process. To compute that measure, we

use the quality parameter q_k of the previous subsection, to also represent the effort spent by editor k each time she edits the document[1]. The total effort is then the sum over all editing slots of that value q_k;

- the repartition of that effort among collaborators, that is quantified by the coefficient of variation of the participant individual efforts during the development process.

C. Results

We now present the performance comparison among the three schemes defined in Section III, focusing on the influence of the size of the collaborative community and the availability of participants.

In each case, we simulated the development of a document a sufficiently large number of times, so that the width of the 90% confidence intervals were at least 100 times smaller than the mean values plotted.

1) Effect of the number of participants:

Convergence time. Figure 1 shows the average number of iterations that are needed before the document reaches a quality that is sufficient to all participants, for the three mechanisms defined before, while Figure 2 compares that convergence time for the trust-based schemes to the one for the naïve round-robin mechanism. As could be expected, the convergence is

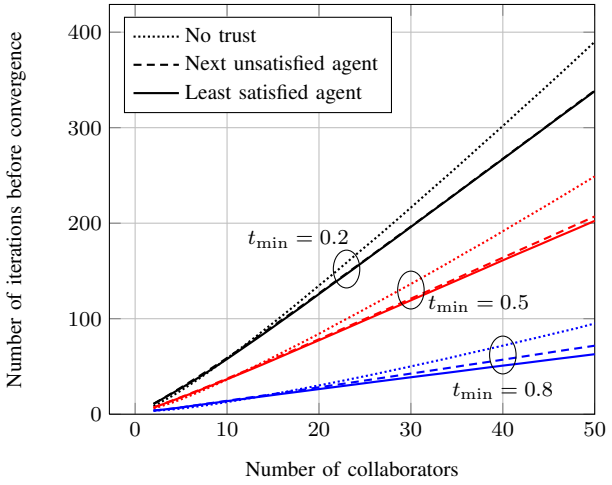

Figure 1. Total time until convergence for the three possibilities of next editor selection.

faster when collaborators trust each other (i.e., when $t_{\min}$ is larger). On the other hand, it unexpectedly appears that the naïve round-robin mechanism is the one that performs best when the number of collaborators is below 10. This is due to the fact that collaborators are assumed to sometimes be unavailable: while in the round-robin case, the documents goes on being improved even if no unsatisfied agent is available, this is not the case for our trust-based schemes and some time slots may be lost. On the contrary, when the number of collaborators becomes large, then such situations occur less frequently, and

[1]In our opinion, it is quite realistic to consider effort and quality to be positively correlated, but the model can be complemented further by adding some randomness.

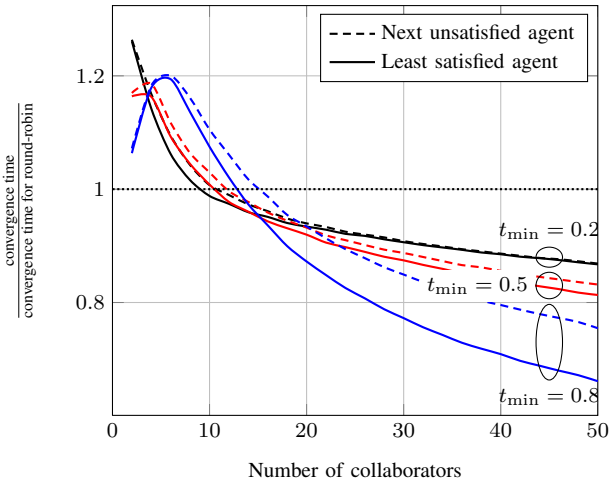

Figure 2. Time gain due to trust-based management.

trust-based mechanisms outperform the round-robin scheme by targeting only unsatisfied agents. The gain in terms of time is for example of 20% for collaborative groups of $I = 50$ participants.

Total effort. Let us now analyze the influence of the participant number I on the total effort spent by the group before reaching a satisfying quality level. Figures 3 and 4 are the respective counterparts of Figures 1 and 2 for that effort performance metric. In terms of total effort, our two trust-

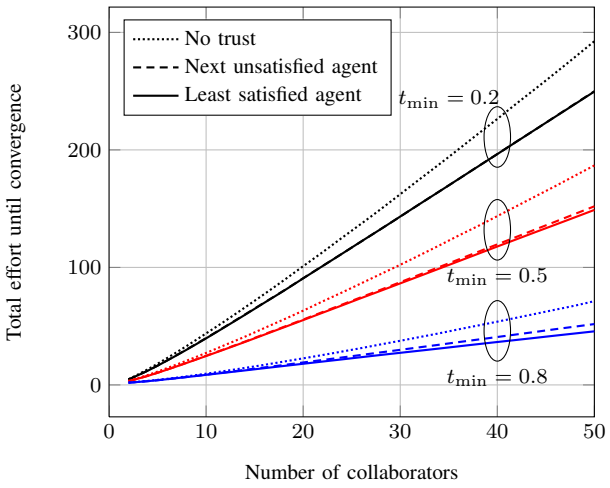

Figure 3. Total effort until convergence for the three possibilities of next editor selection.

based management schemes always yield an improvement with respect to the naïve round-robin mechanism: when $t_{\min} = 0.5$, the total effort is reduced by about 20% for $I = 50$ collaborators, but the improvement is of less than 10% if there are less than 10 collaborators.

Effort repartition. We plot in Figure 5 the variation coefficient of the effort repartition among participants. Simulation results indicate that both trust-based schemes allow a fairer repartition of the total effort than the naïve mechanism.

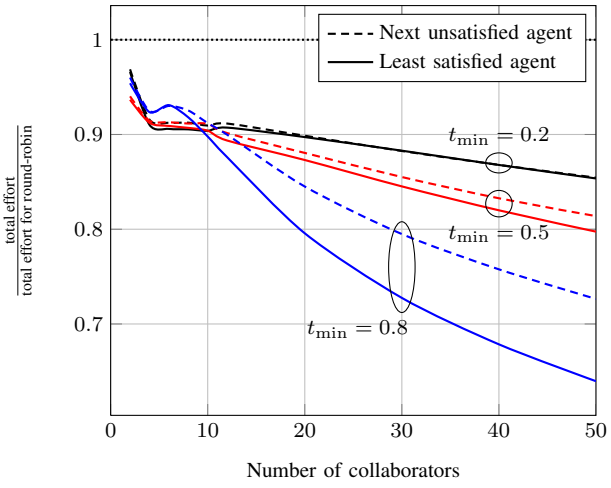

Figure 4. Effort gain due to trust-based management.

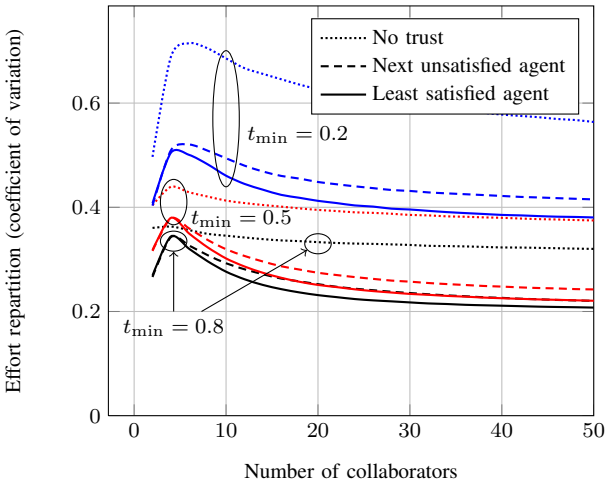

Figure 5. Effort repartition for the three possibilities of next editor selection.

To summarize, the simulations suggest that with few collaborators, introducing trust-based management will not improve the speed of the development process, and the improvement in terms of effort will be quite small. Therefore, in those cases it might not be interesting to use trust. On the other hand, when the number of collaborators becomes large, then applying one of our trust-based schemes yields a non-negligible improvement, in terms of time, of effort spent, and of fairness of the effort repartition. For both performance measures, the difference between the two trust-based schemes is always at the advantage of the one targeting the least satisfied agent, but remains small when compared to the improvement brought by trust. This suggests that the round-robin among unsatisfied agents should be preferred due to the implementation and privacy drawbacks of the other scheme.

2) Effect of the mean availability of participants: We now investigate the influence of participants availability over the development process. More precisely, we vary the probability of each individual user to be available, and observe the consequences on our performance measures. For simplicity reasons, we assume all agents have the same probability of being online. Simulations were carried out for a collaborative working group with $I = 20$ participants.

Convergence time. Figure 6 shows the ratio in terms of convergence time of trust-based mechanisms with respect to the round-robin scheme. Interestingly, we remark that trust-based mechanisms yield an improvement in terms of time only when the participant availability is within an interval (around $[0.3, 0.85]$ for our simulation parameters when $t_{\min} = 0.5$), while for extreme values convergence is faster with the round-robin mechanism. For low values of the availability, this can be explained in terms of time slots losses (i.e., time slots where no agent edits the document), that occur for our trust-based mechanisms and not for the round-robin. For high values of the availability (take the extreme case where all agents are always available for simplicity), the difference may come from the fact that in the round-robin mechanism, all participants (even the satisfied ones) edit the document. There might be some cases where an agent is already satisfied with the document, but has a very high writing quality: in that case, with the round-robin mechanism the agent will improve the quality of the document significantly (thus benefitting to all participants), while she would not have edited it with a trust-based scheme.

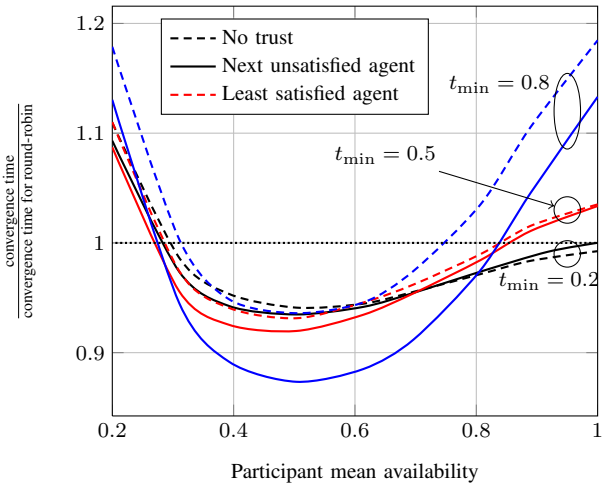

Figure 6. Time gain due to trust-based management.

Total effort. The same phenomenon occurs when considering the effort metric when participants are available very often: the round-robin scheme performs better. However this is not true anymore for low availability values since lost time slots have no impact in terms of effort.

Those results are illustrated in Figure 7, that moreover shows that the two trust-based schemes are almost insensitive to participant availability, while the total cost with the round-robin scheme strongly decreases when participants are more frequently available. As a result, when the availability of participants is not known a priori, using trust enables to still control the total amount of effort needed to reach a satisfying document quality.

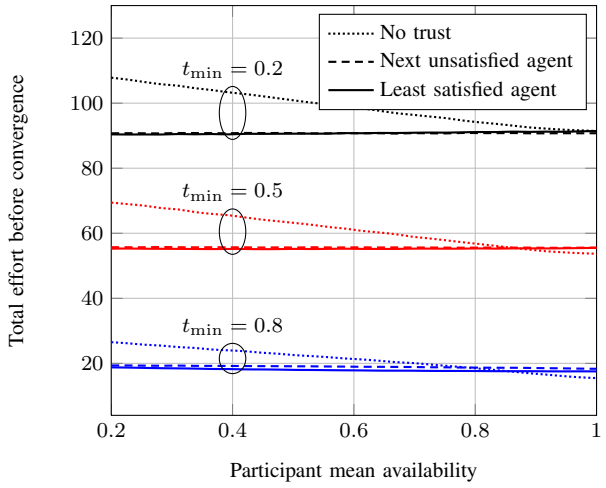

Figure 7. Total effort until convergence

V. Implementation: the case of OpenOffice documents

OpenOffice [www.openoffice.org] is now known as the most popular open-source document editor, with intuitive and familiar interfaces. OpenOffice has been chosen as the principal document editor in many companies, schools, as well as important projects. In order to increase the efficiency of the collaborative working process, we suggest to enable the use of our trust model in the OpenOffice editor using a dedicated plugin that interacts directly with OpenOffice menus. The plugin would be in charge of updating the evaluation records in the metadata (and possibly of storing trust values and computing satisfaction scores).

An OpenOffice text document is in fact a zipped package. An example of such a document is given below:

```
document.odt
├─ styles.xml
├─ settings.xml
├─ meta.xml
├─ content.xml
├─ mimetype
├─ Thumbnails/
├─ META-INF/
└─ Configurations2/
```

The file `meta.xml` contains information about the document itself (metadata information). We propose to use that file to store data computed from our trust-based evaluation model. OpenOffice metadata information rely on six Dublin Core [18] elements (`title`, `subject`, `description`, `creator`, `date`, `language`), and eight own defined metadata elements (`generator`, `initial-creator`, `creation-date`, `keyword`, `editing-cycles`, `editing-duration`, `user-defined`, `document-statistics`).

OpenOffice also allows users to define their own metadata parameters via a user-defined field. We exploit that flexibility by adding four user-defined fields into the metadata of the processed document:

- `transaction_Id`, that is a unique identifier for the transaction (i.e., the last edit);
- `E_last`, the evaluation score set by the last editor;
- `Id_nextEditor`, the identity of the next editor (following the trust-based scheme);
- (in order to have a fully decentralized scheme) `trust_records`: that element stores all trust values between participants in a format `[Id1, Id2,` $t_{\texttt{Id1,Id2}}$`]` that represents the trust value of agent `Id1` on agent `Id2`, for all `Id1, Id2`$\in \mathcal{I}$. Remark that we choose here to regroup all trust values within the document itself (more precisely, in its metadata) so as to prevent the need for extra software or files locally stored by agents. This implicitly means that in a first phase, agents are asked to declare their whole trust vector, which they might be reluctant to do. However, we believe that implementation will be simpler (because it is stand-alone) than other possibilities where trust vectors have to be stored and called locally.
  Since the trust values of the type $t_{k,k}$ always equal 1 for all $k \in \mathcal{I}$, we suggest to use the values $t_{\text{Id1,Id1}}$ in `trust_records` to store the quality threshold b_{Id1} of each user.

Our plugin interacts directly with OpenOffice menus: when a menu item is selected, the corresponding method of our plugin is called instead of the genuine OpenOffice one. When a participant terminates the editing process, she has to select the Export menu which has been re-implemented by our plugin. The editor is then asked to enter an evaluation score that is then stored in the `E_last` field by our plugin. It also generates an identifier for that transaction, that is stored in the `transaction_Id` field. Finally, our plugin computes satisfaction values based on the identity of the last editor, the fields `trust_records`, `transaction_Id`, and `E_last` values; it then chooses the next participant that has to modify the document according to the policy chosen (round-robin among unsatisfied agents, or least satisfied agent). The corresponding participant identity is then stored in the `Id_nextEditor` field. All those metadata are then saved in the `meta.xml` file. Only the elected participant(s) can then edit the document. When all satisfaction values are above the acceptability threshold, the editing process is considered terminated, and our plugin locks the document to prevent further modifications. To prevent malicious modifications and unauthorized access (to trust values for example), all content is encrypted using an X509 certificate. Only our plugin is able to decrypt that document.

The plugin has been developed using Java technology and relies on OpenOffice (v3.1) Java API. Metadata manipulation is made possible thanks to Java library jOpenDocument [www.jopendocument.org], a GPL free library. Bouncy Castle

free libraries [www.bouncycastle.org] have been used for encryption.

VI. Conclusions and future work

In this paper, we have proposed some new management methods, based on the notion of trust, to improve the process of collaborative document writing. Our simulation results show that trust can help reduce significantly the time and effort spent until a satisfying document version is produced, especially in the realistic cases when the number of collaborators is large and collaborators are not always available to edit the document.

We considered a simple model with strong assumptions to highlight some basic phenomena that occur when trust is used in such contexts. However, it would be interesting to extend that model in future work, in different directions. One could be to avoid asking editors to evaluate the document themselves; indeed that can be an incentive to declare an over-estimated score if agents' interests are not perfectly aligned. A possibility would be that the document be checked and evaluated by some collaborators [1], [19]. That would improve the accuracy of the evaluation score, but would also imply some extra evaluation effort.

Also, our simulations rely on a model of user behavior determining the quality improvement of each edit. Since in reality the contribution of each participant is hard to predict, some experiments of real collaborative work (for example with users jointly producing an OpenOffice document using our plugin) could give more insight about the efficiency of our trust-based mechanisms.

The assumption that we make in this paper, of all collaborators having the same objective, is quite strong. We would like to relax it and introduce some objective misalignment among participants, so that the interaction has to be robust to agent selfishness: to ensure that property, the system should be studied as a noncooperative game [20], for which the agent equilibrium strategies should lead to a satisfying outcome.

As another research direction, we would also like to consider the possibility for agents to update their trust vector during the document development process, based on their perception of the contributions from their peers, as is suggested in [21] to detect malicious agents.

Finally, the probabilistic interpretation of the satisfaction value computation could be continued and enriched, in two directions:

- that computation could be based not only on the last version but also on the previous ones (e.g., if some participants do not know the last editor but highly trust the previous one),
- the trust values could be multidimensional, for example following the representation introduced in [22].

Acknowledgements

This work has been partially funded by the French *Agence Nationale pour la Recherche* through the FLUOR project.

References

[1] P. B. Lowry, J. F. Nunamaker Jr, A. Curtis, and M. R. Lowry, "The impact of process structure on novice, virtual collaborative writing teams," *IEEE Transactions on Professional Communication*, vol. 48, no. 4, p. 341, Dec 2005.

[2] P. B. Lowry, J. F. Nunamaker Jr, and M. C. Hall, "Using Internet-based, distributed collaborative writing tools to improve coordination and group awareness in writing teams," *IEEE Transactions on Professional Communication*, vol. 46, no. 4, pp. 277–297, 2003.

[3] T. Grandison and M. Sloman, "A survey of trust in internet applications," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 4, pp. 2–16, 2000.

[4] X. Ding, W. Yu, and Y. Pan, "A dynamic trust management scheme to mitigate malware proliferation in P2P networks," in *Proc. of IEEE ICC*, Beijing, PR China, 2008.

[5] X. Li and H. Zhao, "A personalized group trust management system for collaborative services," in *Proc. of 17th International Conference of Computer Communications and Networks (ICCCN'08)*, St. Thomas, US Virgin Islands, Aug 2008.

[6] Q. Zhang, T. Yu, and K. Irwin, "A classification scheme for trust functions in reputation-based trust management," in *Proc. of ISWC Workshop on Trust, Security, and Reputation on the Semantic Web*, Hiroshima, Japan, Nov 2004.

[7] Q. Zhang, Y. Zhuo, and Z. Gong, "A trust inspection model based on society behavior similarity rule in dynamic networks," in *Proc. of International Conference on Computer Science and Software Engineering*, Wuhan, PR China, Dec 2008, pp. 970–973.

[8] S. Ruohomaa, "Trust management for inter-enterprise collaborations," *Web proceedings of the I-ESA*, vol. 7, 2007.

[9] X. Zheng, P. Maillé, C. T. Phan Le, and S. Morucci, "Trust mechanisms for efficiency improvement in collaborative working environments," in *Proc. of MASCOTS*, Miami, FL, USA, Aug 2010.

[10] J. Golbeck and J. Hendler, "Inferring binary trust relationships in web-based social networks," *ACM Transactions on Internet Technology*, vol. 6, no. 4, pp. 497–529, 2006.

[11] B. Khosravifar, J. Bentahar, M. Gomrokchi, and R. Alam, "An approach to comprehensive trust management in multi-agent systems with credibility," in *Proc. of 2nd International Conference on Research Challenges in Information Science (RCIS)*, Marrakech, Morocco, Jun 2008, pp. 53–64.

[12] S. Marti, "Trust and reputation in peer-to-peer networks," Ph.D. dissertation, Stanford University, May 2005.

[13] G. Suryanarayana and R. N. Taylor, "A survey of trust management and resource discovery technologies in peer-to-peer applications," University of California, Irvine, Tech. Rep. UCI-ISR-04-6, Jul 2004.

[14] A. A. Selçuk, E. Uzun, and M. R. Pariente, "A reputation-based trust management system for P2P networks," in *Proc. of IEEE International Symposium on Cluster Computing and the Grid (CCGRID'04)*, Washington, DC, USA, 2004, pp. 251–258.

[15] S. Javanmardi and C. V. Lopes, "Modeling trust in collaborative information systems," in *Proc. of 3rd International Conference on Collaborative Computing (CollaborateCom)*, New York, NY, USA, Nov 2007.

[16] C. T. Phan Le, F. Cuppens, N. Cuppens, and P. Maillé, "Evaluating the trustworthiness of contributors in a collaborative environment," in *Proc. of TrustCol 2008*, Orlando, FL, USA, Nov 2008.

[17] Y. Liu, "Trust-based access control for collaborative system," in *Proc. of ISECS International Colloquium on Computing, Communication, Control, and Management*, Guangzhou City, China, Aug 2008.

[18] S. Weibel, "The Dublin core: A simple content description model for electronic resources," *Bulletin of the American Society for Information Science*, vol. 24, no. 1, pp. 9–11, 1997.

[19] S. Sumiya and T. Saito, "Development of a multimedia document management system for cooperative work environment," in *Proc. of 16th Computer Software and Applications Conference (COMPSAC)*, Chicago, IL, USA, Sept 1992, pp. 346–355.

[20] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. MIT Press, 1994.

[21] K. S. Barber and J. Kim, "Soft security: Isolating unreliable agents from society," in *Proc. of the International Workshop on Deception, Fraud, and Trust in Agent Societies*, Bologna, Italy, Jul 2002.

[22] A. Jøsang, "Artificial reasoning with subjective logic," in *Proc. of 2nd Australian Workshop on Commonsense Reasoning*, Perth, Australia, Dec 1997.