



# UNIVERSITÀ DEGLI STUDI DI TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

---

38050 Povo — Trento (Italy), Via Sommarive 14  
<http://dit.unitn.it/>

WIKIREP:  
DIGITAL REPUTATION IN VIRTUAL COMMUNITIES

Mikalai Sabel, Anurag Garg, Roberto Battiti

June 15, 2005

Technical Report # DIT-05-050



# WikiRep: Digital Reputation in Virtual Communities

Mikalai Sabel, Anurag Garg, Roberto Battiti  
Dipartimento di Informatica e Telecomunicazioni,  
Università degli Studi di Trento  
{msabel,garo,battiti}@dit.unitn.it

June 15, 2005

## Abstract

We present a framework for applying digital reputations to collaborative environments, where multiple authors contribute to the same document. In our mechanism readers evaluate a single document, and reputation data derived from the feedbacks affect multiple authors and document versions. The digital reputation encourages contribution and provides motivation for active participation, leading to better quality of the content and to better user experience. Our approach is applicable to user-managed communities, distributed moderation systems, and other collaborative contexts.

The underlying ideas of our scheme are: separate ratings for different versions of the same document, inheritance of previous ratings for newer versions, and appropriate allocation of credit to versions and authors. We describe in detail both the algorithms used to compute reputations, and our implementation, WikiRep, a reputation-enabled Wiki application.

## 1 Introduction

The aim of this work is to use digital reputations in a collaborative virtual environment. The concepts of digital trust and reputations are popular and widely used in online communities today, for example, in *slashdot* (slashdot.org), *epinions* (epinions.com), *ebay* (ebay.com) and *amazon* reviews (amazon.com). However, available systems assume that each object (article, review, transaction) has a single responsible person (author, seller), and that

the objects are independent from each other. A typical scenario for a system like *slashdot* is presented on Figure 1. The system consists of community members (i.e. users of the system), who can act like authors or readers, and objects — articles and comments. First, an author creates an object, then a reader evaluates it and leaves a feedback. From the feedback, the system derives a rating for the object, a reputation of the author, and a personalized opinion of the author by the reader. For each object, there is only one author and many readers. Analysis and experiments on existing applications have shown that such distributed moderation is feasible in practice [8, 2, 12].

We present a generalization of the reputation-based approach, see Figure 2, where there are several authors creating different versions of the same document. Readers read and evaluate the document as a whole, and the feedback is allocated to the authors and the versions. Some issues concerning users' behavior in collaborative environments have been discussed in the context of *Collaborative programming contest* [6], but the scenario is restricted: instead of reviews from community, there is an objective metric for object's value (program execution time), and only the last author is rewarded, independently of degree of his contribution. In this paper, we present an approach to assign credit to the different authors starting from the feedback values, and to derive digital opinions in a generalized multi-authored scenario.

As the implementation platform, we have chosen Wiki, a highly collaborative virtual application for writing online documents. Wiki has achieved significant popularity during recent years, and collaboration is one of its key features. Other examples of collaborative applications are: co-authorship in writing documentation, team software development, online discussions, etc.

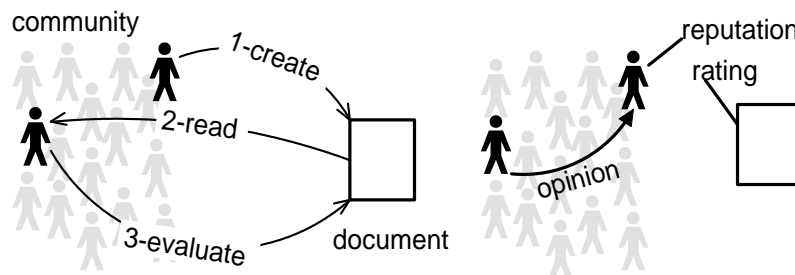


Figure 1: Context for digital reputations: single document and single author

The motivation for introducing digital opinions is that they automate maintenance of the virtual environment, and they require less dedicated human resources [1]. The systems become more scalable and affordable for a larger range of environments. In addition, digital opinions provide motivation for active participation, by rewarding the best contributors with increased

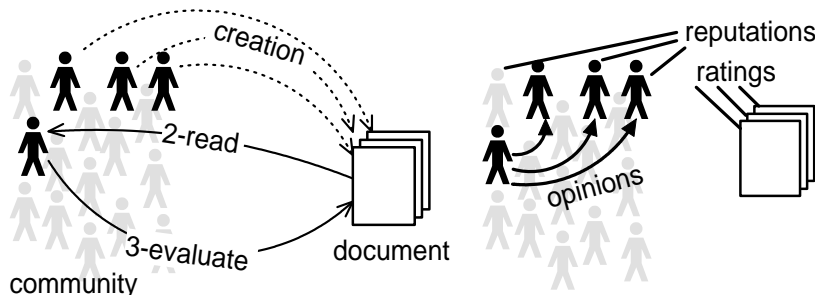


Figure 2: Context for digital reputations: collaborative authors and document with internal version structure

reputation to indicate their achievements. The major challenges in employing reputations in a collaborative scenario are:

- The highly dynamic nature of content: each object is modified many times during the collaborative editorial process, and evaluations may become irrelevant. To solve the problem, we suggest to apply evaluations to individual versions, and to use *rating inheritance* from already evaluated documents to newer versions.
- Each object contains contributions of many members. Feedback has to be processed differently to derive ratings and opinions for all reused versions and their authors. For that, we use *author* and *version allocation factors*.

In this paper we concentrate on mechanisms for deriving opinion data from feedback in a Wiki collaboration environment. Section 2 describes the case study context (Wiki), and the assumptions made. Section 3 presents the design of the feedback allocation procedure, and Section 4 describes prototype implementation, including algorithmic details and design choices.

## 2 Context, definitions and assumptions

In this Section, we briefly describe the case study we consider (Wiki collaborative environment), provide definitions of reputations, opinions and ratings, and explain the assumptions about the page structure and opinions.

The Wiki concept and the first implementation were introduced by Ward Cunningham in 1995 [13]. Wiki is a method to build and maintain a web-site with pages written mostly by visitors. Many Wiki implementations have been created (MediaWiki [7], Twiki, etc.), but to the best of our knowledge

none of them employs digital opinions and reputations to improve usability and content quality. If different access permissions are supported, usually the administrator or the first page author manually defines access rights. As result, it takes considerable time and effort to perform administration, and the whole community relies on the activity of particular members.

We consider our opinion-handling mechanism as an add-on to the base system (Wiki in our case). Figure 3 displays the resulting system, with added elements marked in grey. The new elements are: storage structures for rep-

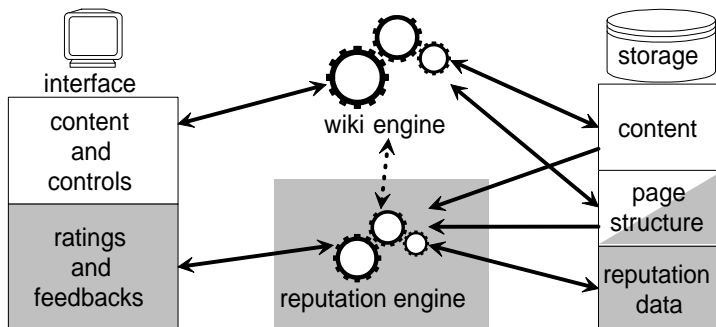


Figure 3: System overview: Wiki + digital opinions

utation data and for additional information about pages structure, interface elements to visualize reputations and collect feedbacks, and an engine to process reputation data and interact with the basic Wiki engine. The interaction between the basic system and the reputation engine (dotted line) corresponds to the various usages of the collected data. A straightforward example is opinion-based filtering: pages having a low rating are not displayed to the readers. In general, the reader must be given a high level of flexibility in deciding about how to use the reputation system. Our focus is on the creation of reputation data, and the possible ways to use these data are outside of the scope of this paper.

**Opinions, reputations and ratings.** From a user’s point of view, *opinion* is a perception about an entity, based on previous experience. Equivalently, it is a collection of evaluations used to build predictions. The evaluations may have varying impact on community. A local (personalized) opinion is used by its originator and nobody else, while a global opinion affects all community members [9]. Global opinions are also called *reputations* or *ratings*, when they refer to community members or objects, respectively. With personal opinions, each member controls his own view of the community, and the damage due to abuse is minimized. However, to collect enough personal feedbacks, the interactions must be repetitive, and an initial ‘bootstrap’ period

is needed. At the same time, global reputations and ratings reuse previous evaluations made by all community members, and are available even for novice participant or external visitors, who do not have sufficient personal interaction experience. The disadvantages of global reputations are the risk of cheating and the lack of customizability for individual users. The feedback allocation mechanism described in Section 3 may be used equally for both personal and global opinions.

When considering the technical aspects of opinion computations, we assume that feedbacks are processed as pairs  $(value; quality)$ , where *value* defines the feedback mark, i.e., reader’s evaluation of object’s merit, and *quality* characterizes significance of the evaluation. *Quality* is a weight normalized to the  $[0, 1]$  interval, with 0 meaning a completely unreliable evaluation that is not considered, and 1 corresponding to a reliable evaluation. Details of the representation we use are explained in Section 4.

**Page structure.** Successive versions of the same page normally have similar content, because the newer versions are modifications of the older ones. The simplest model of a Wiki page is a sequence of versions, ordered by their time of creation. Each version, except the very first, is a descendant of the preceding one. Unfortunately, this model is insufficient to describe a realistic evolution of a Wiki page, when contributors perform reverts and re-insertions from older versions. The most general model of a Wiki page is the one in which all previous versions are considered as contributors, which results in high complexity.

A feasible tradeoff between simplicity and generality is achieved by tree model (Figure 4). In this case, each version has at most one *parent* version. The parent is considered to be the major foundation for its child version(s). If a version is completely original (e.g. re-written from scratch, or the first version), it has no parent. The tree model is reasonably simple and captures the typical behavior of a Wiki user well: a new version is normally created starting from one of the existing versions, but not necessarily the latest. Hereafter, we assume that the tree model is the one employed.

Relations between pages and their parents are characterized numerically by *adoption coefficients*. Consider two versions  $i$  and  $j$  of the same page,  $i$  being older than  $j$ . The *adoption coefficient*  $a_{i,j}$  characterizes the ‘similarity’ between the versions, in particular, it measures how much content of version  $i$  is preserved in version  $j$ . The scale for adoption coefficient is  $[0, 1]$ , with 0 corresponding to independent versions, and 1 meaning that  $j$  is a copy of  $i$ , with no differences. Generally, any automated algorithm for producing adoption coefficients is feasible, starting from naive text comparison and

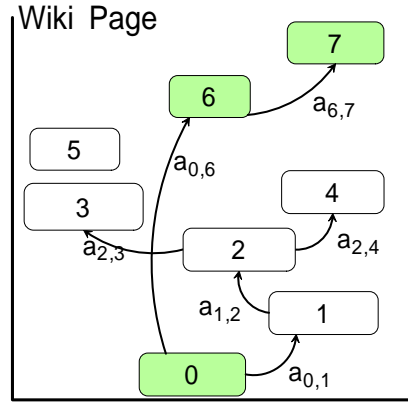


Figure 4: Tree model of Wiki page

up to semantic-aware tools, depending on requirements, resources and definition of ‘similarity’ in particular environment. An approach that we have implemented is presented in Section 4.

For the tree page model, we denote parent of version  $j$  as  $parent(j)$ . For example, on Figure 4,  $parent(7) = 6$ . Because each page version has at most one non-zero adoption coefficient (the one towards its parent), a simplified notation  $a_j := a_{parent(j),j}$  is used for adoption coefficients. In fact, since adoption coefficients are the metric for versions’ similarity, in our system the parent version is chosen as the version having the maximum adoption coefficient.

### 3 Collecting reputation data

The major novelty of our approach is its capability to handle objects consisting of *several versions* and created by *different authors*. This section describes the algorithms to maintain reputations and opinions in such environment.

**Rating inheritance.** When a new version of a page is created, it partially inherits the rating of its ancestors. To be coherent with the feedback allocation procedure below, a new version inherits fraction  $a_j$  of its parent’s rating. If the parent’s rating is  $(R_{parent(j)}; quality_{parent(j)})$ , the initial rating of the newly created page  $j$  is initiated with  $(R_{parent(j)}; a_j \cdot quality_{parent(j)})$  pair, which means that value of the rating stays the same, but its reliability decreases. Only fraction  $a_j$  of the content is inherited, and so is rating weight. The other part,  $(1 - a_j)$ , is new and not evaluated yet, so it is considered to

have zero reliability weight.

**Feedback allocation.** Each reader can leave a feedback about page that he reads as a single simple action, e.g. pressing (+) or (-) button. The feedback affects not only the version being viewed, but also the previous versions and their authors. Each involved page version and author receives and individually collects a ‘fraction’ of the feedback given by the reader.

When a page version  $n$  receives feedback evaluation mark  $M$ , the version  $n$  itself adds pair  $(M; w_{n,n})$  to its version rating and  $(M; u_{n,n})$  to its author’s reputation. The addition actually is a weighted summation, and a weighted average of the evaluations is used later as the opinion value. Version  $m = \text{parent}(n)$  adds  $(M; w_{m,n})$  to its rating and  $(M; u_{m,n})$  its its author’s reputation, and so on. Each ancestor  $i$  of  $n$  receives feedback  $(M; w_{i,n})$  for the page version and  $(M; u_{i,n})$  for its author. Weights  $w_{i,n}$  and  $u_{i,n}$  depend on the page structure, and are defined and explained below. After the feedback weights are defined, the opinions about each version and each version’s author are updated. Each feedback pair is added to a corresponding opinion, reputation or rating, using value  $M$  and the proper weight.

**Feedback allocation to versions.** Feedback is allocated to page versions using *version allocation factors*  $w_{i,j}$ . Page rating characterizes the page version as a whole, and not just the added value beyond its ancestors. *Version allocation factor*  $w_{j,j}$  for a version  $j$  itself is 1, i.e. the version which has attracted the feedback from the reader gets the ‘full’ weight, because it is what the reader expects when giving the feedback for this version. Parent version of the evaluated version gets  $a_j$ -weighted evaluation, because  $a_j$  part of the read content was actually taken from the parent. This continues for all page versions that are ancestors of  $j$ :

$$w_{i,j} = \begin{cases} 0 & \text{if } i \notin \text{contributors}(j) \\ 1 & \text{if } i = j \\ \prod_{k \in \text{inter}(i,j)} a_k & \text{if } i \in \text{contributors}(j) \text{ and } i \neq j \end{cases}, \quad (1)$$

where  $\text{contributors}(j)$  refers to the chain of all versions that are re-used in  $j$ , including  $j$  itself:  $j, \text{parent}(j), \text{parent}(\text{parent}(j))$ , and so on, until a version without any parent is finally encountered. The set  $\text{inter}(i,j)$  contains  $j$ ’s ancestors between  $j$  and  $i$  (excluding  $i$  and including  $j$ ):

$$\text{inter}(i,j) = \text{contributors}(j) \setminus \text{contributors}(i) \quad (2)$$

Each *version allocation factor* includes both the part of feedback corresponding to the added value of the given version and part corresponding to all its ancestors.

**Feedback allocation to authors.** Each coefficient for allocating feedback to version's author, *author allocation factor*, measures original contribution added in the corresponding version. The factor is defined as:

$$u_{i,j} = \begin{cases} 0 & \text{if } i \notin \text{contributors}(j) \\ 1 - a_j & \text{if } i = j \\ (1 - a_i) \prod_{k \in \text{inter}(i,j)} a_k & \text{if } i \in \text{contributors}(j) \text{ and } i \neq j \end{cases}, \quad (3)$$

The coefficients  $u_{i,j}$  possess a useful property: *Sum of coefficients  $u_{i,j}$  between a given page version  $j$  and all other versions is 1:*

$$\sum_i u_{i,j} = 1, \text{ for any } j \quad (4)$$

*Proof.* We provide a proof by induction.

*Basis:* consider version  $m$ , such that  $a_m = 0$ , i.e.  $m$  has no parent. For  $m$ , there is only one non-zero coefficient  $u_{m,m} = 1$ . Thus, the proper sum is 1, and (4) holds.

*Induction hypothesis:* assume that (4) holds for some page version  $k$ .

*Induction step:* consider version  $l$ , such that  $k = \text{parent}(l)$ , i.e.  $l$  is  $k$ 's child. Then (4) holds for  $l$  as well:

$$\sum_i u_{i,l} = u_{l,l} + \sum_{i \neq l} u_{i,l} = 1 - a_l + a_l \sum_{i \neq l} u_{i,k} = 1 - a_l + a_l \sum_i u_{i,k} = 1 \quad (5)$$

We have used (3) to isolate  $a_l$  multiplier, then the fact that  $u_{l,k} = 0$  to generalize the sum, and finally substituted the sum using (4) for version  $k$ .

So, we conclude that (4) holds for all page versions.  $\square$

The property ensures that a user can never receive additional reputation by creating superfluous versions.

Figure 5 schematically outlines the feedback allocation process for the example page from Figure 4.

## 4 Implementation

Our implementation is based on eGroupWare [3], a free software to support group activities, that contains a set of applications, including Wiki. The Wiki implementation is originally based on WikiTikiTavi engine (tavi.sourceforge.net). The modifications we introduced are: calculating and storing adoption coefficients and parent references, calculating and storing of opinions, interface elements to visualize reputations and input feedbacks.

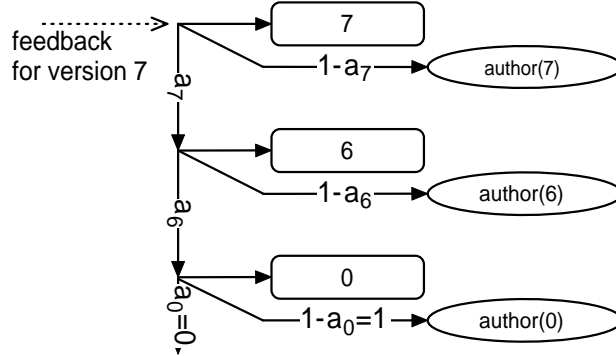


Figure 5: Feedback allocation factors and adoptions coefficients

**Adoption coefficients.** Whenever a new version of a Wiki page is saved, adoption coefficients to all previous versions are calculated, the largest coefficient determines the parent of the version and is stored for later use. There are three properties of a Wiki page version that can be used to compute adoption coefficients: author, modification time, and page text itself. We use a text comparison to measure similarities and differences between page versions.

Adoption coefficients are calculated in the following way. Texts of two versions to compare are divided into *blocks*, using as separators punctuation marks (full stops, commas, colons, semicolons, exclamation and question marks). Consecutive spaces and new lines are ignored. Then, Levenshtein distance (edit distance) between the texts is found, using the blocks as characters [10]. The distance is the minimal number of insert, modify and delete operations on characters, needed to transform one text into the other:  $(N_{inserted} + N_{modified} + N_{deleted})$ . The distance is then normalized to fit into  $[0, 1]$  range by division by the maximum possible value of the distance:  $(N_{total,new} + N_{deleted})$ , where  $N_{total,new}$  is the total number of blocks in the newer version. Finally, subtracting this normalized distance from 1 gives the adoption coefficient:

$$a_{old,new} = 1 - \frac{N_{inserted} + N_{modified} + N_{deleted}}{N_{total,new} + N_{deleted}}, \quad (6)$$

**Opinions and reputations in our system** Considering the tradeoff between purely local opinions and global community-wide reputations, we choose a combination of both approaches, in order to fit different communities, members, and usage scenarios. Two types of opinions are stored in the system: local member-about-member opinions, and global object reputations (*page ratings*). Besides that, global user reputations are derived from local

member-about-member opinions. Use of the reputations is restricted to cases when local opinions are not sufficient or not applicable. Local user-about-page opinions are not used explicitly, but each user can evaluate each page version only once.

Opinion representation in our system is based on the ROCQ (Reputations, Opinion, Credibility and Quality) scheme [4, 5]. All evaluations (including opinions, reputations and ratings) are expressed as real numbers in  $[0, 1]$  range, with 0 being the lowest mark, and 1 being the highest. A second value, *quality*, is used to characterize significance of the corresponding evaluation. Quality is also a real number within  $[0, 1]$  interval, with 0 meaning a completely untrustworthy value, and 1 corresponding to a reliable value. Altogether, opinion or reputation is a pair (*value*; *quality*).

Each opinion is derived from a set of individual marks, gathered during previous interactions or coming from other sources. The set of marks is treated as a random sample from independent identical distributions. Sample mean (weighted average) gives value of the opinion:

$$value = \frac{S}{N}, \quad (7)$$

where  $S$  is the weighted sum of all collected evaluations marks, and  $N$  is the weighted number of the marks (sum of their weights).

*Quality* is derived as probability that interval  $[value - \Delta_r/2; value + \Delta_r/2]$  holds the actual mean of the underlying mark distribution. Width of the interval  $\Delta_r$  is a parameter to be chosen. The difference between actual and estimated means has Student's t-distribution [11] with  $\nu = N - 1$  degrees of freedom. Overall, *quality* is found as:

$$quality = 1 - I_{(N-1)/(N-1+t^2)} \left( \frac{N-1}{2}, \frac{1}{2} \right), \quad (8)$$

where  $I_x(a, b)$  is incomplete beta function, and

$$t^2 = \left( \frac{\Delta_r}{2} \right)^2 \frac{N^2(N-1)}{N \cdot S_q - (S)^2}, \quad (9)$$

where  $S_q$  is the weighted sum of squares of collected evaluations marks. More details on calculation of  $I_x(a, b)$  and source code that was adopted in the implementation are found in [11]. Parameter  $\Delta_r$  in our system is constant and fixed at 0.1 level, slightly different from ROCQ, where it is a fraction of *value*, i.e.  $\Delta_r = r \cdot value$  (where, for example,  $r = 0.1$ ).

The algorithm above cannot be used for  $n \leq 1$ , and provides meaningful results only for sufficiently large  $n$ , at least  $n > 2$  is expected. Therefore,

when  $n$  is small, either opinion quality is treated as ‘not defined’, or default initial quality is used.

We suppose that the modifications made to not damage the ease of use that Wiki provides. Only two elements are added to the page visualization interface: symbolic representation of page rating and buttons to leave feedback (Figure 6).

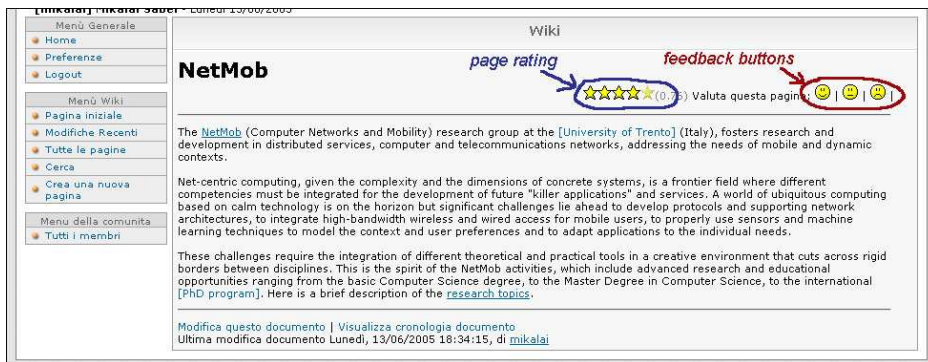


Figure 6: WikiRep interface screenshot

## 5 Conclusions

In this work, we extend reputation algorithms for user-managed communities and distributed moderation, by providing a mechanism for deriving reputations and opinions in a collaborative environment with multi-authored, multi-versioned objects. Our approach is applicable to different contexts, such as co-authored writing, documentation and source code development, and online discussions. Successful deployment of reputations in these environments will increase the quality of contributions, and provide motivation for users to participate more actively.

The algorithm relies on maintaining reputations of individual pages and properly allocating of feedback credit. Already established page reputations are reused via *rating inheritance*. Reputations for authors and page ratings are treated consistently, but with different *allocation factors*, reflecting their different meanings.

We implemented the algorithms by using Wiki as our base system. Additional data for reputations, opinions, and pages structure are stored and handled by the reputation engine. The system is currently deployed, and experiments and observation of a real virtual community are already under way.

## Acknowledgments

This work is funded by Provincia Autonoma di Trento through the WILMA Project.

## References

- [1] Meatball Wiki: a community of communities. <http://www.usemod.com>.
- [2] Dan Cosley, Dan Frankowski, Sara Kiesler, Loren Terveen, and John Riedl. How oversight improves member-maintained communities. In *CHI '05: Proceeding of the SIGCHI conference on Human factors in computing systems*, pages 11–20, New York, NY, USA, 2005. ACM Press.
- [3] eGroupWare. <http://www.egroupware.org/>.
- [4] Anurag Garg and Roberto Battiti. The Reputation, Opinion, Credibility and Quality (ROCQ) scheme. Technical Report DIT-04-104, Informatica e Telecomunicazioni, University of Trento, 2004.
- [5] Anurag Garg, Roberto Battiti, and Gianni Costanzi. Dynamic self-management of autonomic systems: The Reputation, Quality and Credibility (RQC) scheme. In *To appear in The 1st IFIP TC6 WG6.6 International Workshop on Autonomic Communication (WAC 2004)*, October 2004.
- [6] Ned Gulley. In praise of tweaking: a wiki-like programming contest. *Interactions*, 11(3):18–23, 2004.
- [7] Wikimedia Foundation Inc. Wikipedia, the free encyclopedia. <http://en.wikipedia.org/>.
- [8] Cliff Lampe and Paul Resnick. Slash(dot) and burn: distributed moderation in a large online conversation space. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 543–550, New York, NY, USA, 2004. ACM Press.
- [9] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. Notions of reputation in multi-agents systems: a review. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 280–287, New York, NY, USA, 2002. ACM Press.

- [10] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.
- [11] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C, 2nd. edition*. Cambridge University Press, 1992.
- [12] Paul Resnick, Richard Zeckhauser, Eric Friedman, and Ko Kuwabara. Reputation systems: Facilitating trust in internet interactions. *Communications of the ACM*, 43(12), Dec 2000.
- [13] Wiki Wiki Web. <http://c2.com/cgi/wiki>.