WIKIREP: INSTALLATION GUIDE AND DEVELOPER'S MANUAL

Mikalai Sabel, Anurag Garg, Roberto Battiti

December 30, 2005

Technical Report # DIT-05-085

WikiRep: Installation Guide and Developer's Manual

Mikalai Sabel, Anurag Garg, Roberto Battiti

December 30, 2005

Abstract

This report covers installation, internal organization and algorithms of WikiRep, an environment for virtual communities with support for digital reputation. The first part is technical, it describes WikiRep installation procedure, briefly outlines source code structure, listing the most important functions, and provides details on how virtual reputations are stored in the database.

Later part of the report covers WikiRep algorithms. Two approaches to defining quality of a digital reputation are analytically compared. The method employed for calculating adoption coefficients is described, with the proof that the coefficients are symmetric. Finally, the approach used to visualize *value* and *quality* of digital reputations is explained in detail.

1 Installation of WikiRep

WikiRep is a Wiki with support for digital reputations and ratings. The implementation we develop is based on the Wiki application in eGroupWare. eGroupWare (eGW) is a web application to support group activities and interactions. In this Section, we describe the process of WikiRep installation, the major modifications WikiRep makes to eGW, and necessary information about WikiRep and eGW architecture. For more information about eGroupWare, and its development, refer to *eGroupWare website* [2], as a starting point we suggest *Coding Standards* [5] and *Style Guide* [6].

WikiRep version 1.0 has been developed and tested with eGroupWare versions 1.0.0.006, 1.0.0.009-2, and 1.0.0.009-3, which is the last release available as this document is prepared. Our platform is Linux, MySQL-4.1.10, Apache-2.0.53, PHP-4.3.10, mod_ssl-2.0.53, OpenSSL-0.9.6i.

Documentation for the source code is maintained with use of an automated tool, *phpDocumentor* [8]. This is a widely used auto-documentation tool for PHP language. It generates the documentation from source files using PHP language structure and specially-formatted comments. An example of generated documentation is found at *phpdoc for eGW and WikiRep* [14], see the descriptions of file `rocq_functions.php` and also of the class `soWikiPage`.

To install WikiRep, you need to install eGroupWare first. After you setup eGroupWare, you apply WikiRep installation, which patches PHP files and modifies the database. You can significantly improve security of the system by configuring SSL connections. Finally, you can customize the system for your particular usage scenario. This Section gives a brief tutorial about these steps.

To install eGW, you should already have installed and setup a compatible web-server with support for PHP, and a database. The steps below were tested on a Linux-operated PC with MySQL database (version 4.1.10) and Apache webserver (Apache/2.0.53 (Unix) mod_ssl/2.0.53 OpenSSL/0.9.6i PHP/4.3.10). Compatibility with other back-end software is not confirmed.

1.1 Installation of eGroupWare

Below is a brief description of installation and initial setup of eGroupWare. For more details, refer to eGW documentation, for example *How To Install and Secure eGroupWare* [11]. We have tested versions 1.0.0.006 and 1.0.0.009-3 of eGW to be compatible with WikiRep.

eGroupWare (eGW) is written in PHP programming language, and uses a database to store its main data. Settings are stored in the file `/header.inc.php` and in the database. Hereafter, all path are given relatively to the main eGroupWare installation directory.

To run, eGroupWare obligatory needs a web-server (e.g. Apache) with PHP support and a database server (e.g. MySQL). Additional applications may depend on other resources, such as filesystem or mail server [7]. We suggest the following steps:

1. Unpack eGW files to a directory in webserver's document root.

2. If necessary, change the file permissions, so that all eGW files are owned by the same user the webserver runs under (for example, use a command like `chown -R apache:apache /var/www/egw`).

3. Create a database (DB) and a database user account. For example, you can do it by running the following script in MySQL (change the DB name, the username and the password):
```
CREATE DATABASE wikirepdb;
GRANT ALL PRIVILEGES ON wikirepdb.* TO 'wikirepuser'@'localhost' IDENTIFIED
BY 'paSSw0rd';
```

4. Go to the web-page corresponding to the home of eGW. There you will find web-interface for initial setup.

   (a) *Choose the language* for setup interface.

   (b) *Run installation tests.* Read all messages about errors and warnings, fix their sources if necessary.

   (c) *Continue to the Header Admin.* (*Header admin*, also called *header manager* is the ultimate eGW administrator, responsible for setup of database, session types, and accounts of other administrators.) Enter server root path, if current is not correct. Setup username and password for the *Header Admin*, it will be used in future to access these settings. Setup the database connection: *DB Name*, *DB User*, and *DB Password.* Setup *Configuration User* account, also called *Setup/Config Admin* — an administrator who installs eGW applications. Finally, press *write config*, or use another way to save the settings to the `/header.inc.php` file.

   (d) Login as *Setup/Config Admin* (Configuration User). *Install applications*, this populates the database. Perform the next step of *configuration* (URLs, paths, authentication, etc.) Create *Admin Account*, it will be used to manage users, applications, interface, and so on. (This is different from *Header Admin* and *Setup/Config Admin*).

   (e) Go *back to user login.*

5. Now, it is a suitable moment to install WikiRep (see below), and then continue to configure eGW applications.

6. Finally, login as admin (*not* Header or Setup/Config admin). Configure available applications, users, groups, preferences, and so on.

1.2 Installation of WikiRep

The WikiRep is installed as a patch for eGroupWare.

1. Unpack *wikirep-1.0.*.tar.gz* archive to the folder eGW is installed to. Some original eGW files will be overwritten.

2. Edit file `/wikirep/index.php`, which comes from the archive: enter to the corresponding fields valid name and password for a database user, and name of the database. The name and the password can be the same that you used for configuring eGW, and the DB name

must be the same. Generally, you may do it before uploading the files to you server, if you have a limited remote access.

3. Open URL 'www.your-egw-site.com/egw-path/wikirep/' in your browser. When you do this, the script in `/wikirep/index.php` executes SQL queries from the file `/wikirep/wikirep.sqls`. You can execute the script or queries by any other means.

4. **Delete** the folder '/wikirep/'. This step is important. The directory is only used during setup, and being left on your server, it creates a severe security risk.

1.3 Setting up secure SSL/TLS connections for registration and login

To improve security of the system, registration and login processes may be handled through a encrypted SLL connection. Below is a brief outline of the suggested steps (all of them except the last two are settings of the web-server):

1. Setup an HTTPS server, prepare and install a certificate for it.

2. Allow only secure connections to pages `login.php` and `registration/*`.

3. Setup a redirect from `'http://www.your-egw-site.com/egw-path/login.php'` to `'https://www.your-egw-site.com/egw-path/login.php'`.

4. In the eGW setup (accessed through the web-interface as config admin), you need to enter the *full* URL of the installation, e.g.
   `http://www.your-egw-site.com/egw-path`

5. Finally, in the `header.inc.php` file in eGW root directory, add line
   `$GLOBALS['phpgw_info']['ssl_loginreg'] = True;`
   This is *not* enabled by default, and this tells the eGW that registration links should use '`https://`' prefix.

The described above steps have been tested for Apache web-server with mod_ssl extension.

2 WikiRep Internals

This section describes new and modified parts of Wiki code that add support for reputations and ratings. Explanation of the reputation mechanisms is found in *DIT Technical Report #05-050. WikiRep: Digital Reputation in Virtual Communities* [15]. Technical details are given in the auto-generated documentation [14] and directly in the source code comments.

2.1 Wiki in eGroupWare

Original eGW Wiki is based on WikkiTikkiTavi wiki engine, and it contains some legacy code, while other parts are (re-)written using new eGW concepts of *phpgwapi* and *eTemplates.* Basically, eTemplate-based parts use *phpgwapi* engine to create interface (for example, dialog pages). Definitions of the interface are stored in the database, created and edited using another eGW application, eTemplates. Older parts of the code create interface from templates stored directly in files. Both approaches have positive and negative points. eTemplates have advantage of 'visual editing' and universality within eGW, but are complex and suffer performance penalty. When implementing new features, we adopted eTemplates methodology, as this is the trend in eGW development. New dialog pages (e.g. 'create page') are based on eTemplates. However, some features are heavily based on previous code, and in those cases static PHP-templates are preserved (e.g. for 'search results' page).

Additional information about eTemplates is found in *'eTemplate: widget based template system for eGW'* [4], *'eTemplate — Templates and Dialog-Editor for eGroupWare'* [3] and in *'eGroupWare Application Development'*(may be outdated) [1].

2.2 WikiRep files and procedures

Significant part of new reputation-processing functions is located in the file `/wiki/lib/rep_functions.php` (this file does not exist in eGW installation). In particular, the important functions are:

calculate_adoption($old_text, $new_text) calculates *adoption coefficient* for two given texts;

triplet2visible($triplet) gives the number of 'stars' for the opinion triplet;

add_duplet($oq, $sums), triplet2pair($sums), triplet2pair_rocq($sums) are used to convert between triplet and pair representations of opinions;

Duplet- and triplet-based opinion representation. So called triplet is a statistical aggregation of all collected feedbacks. It is used for storing and collecting marks. Triplet consists of three real values: sum = sum of marks, $sum2$ = sum of squares of the marks, and n = sum of weights of the marks. So called duplet or pair is the processed opinion representation, used for interpretation and for processing. It consists of two real numbers: *value* and *quality*. For more details, see *WikiRep: Digital Reputation in Virtual Communities* [15], page 10.

Significant modifications have been made to `/wiki/inc/class.sowiki.inc.php`, the class that defines wiki interaction with the database. In particular, the following functions were modified or added:

soWikiPage:read() now reads also rating and reputation data, and can perform filtering when choosing current page version;

soWikiPage:write() now calculates adoption coefficients (AC), determines parent version and saves the parent version's number and corresponding AC;

soWikiPage:detect_parent_version() a new function called from soWikiPage:write();

soWikiPage:process_feedback($mark) a new function that is called whenever viewer provides a feedback. The function performs feedback allocation and updates reputation and rating data;

sowiki:find($text,$search_in=False,$sort_by='nameasc') was modified to support sorting options (including rating-based);

Visualization of ratings and introducing feedback input affected the following interface-related files: `/wiki/action/view.php`, `/wiki/template/view.php`, and `/wiki/parse/html.php`.

Complete list of modified and added files in WikiRep 1.0, compared to eGroupWare 1.0.0, with path given relative to the eGroupWare base path:

```
Registration and login-related files:
./index.php
./login.php
./home.php
./phpgwapi/inc/class.sessions.inc.php
./registration/inc/class.uireg.inc.php
./registration/inc/class.soreg.inc.php
./registration/inc/class.boreg.inc.php
./registration/main.php
./registration/templates/idots/layout.tpl
./registration/templates/default/confirm_email.tpl
./registration/templates/default/config.tpl
./registration/inc/hook_logout.inc.php
./registration/templates/default/loginid_select.tpl
```

```
Modifications to the Wiki:
./wiki/lib/rep_functions.php
./wiki/inc/hook_sidebox_menu.inc.php
./wiki/inc/class.uiwiki.inc.php
./wiki/inc/class.sowiki.inc.php
./wiki/inc/class.bowiki.inc.php
./wiki/lib/defaults.php
./wiki/lib/url.php
./wiki/lib/main.php
./wiki/lib/diff.php
./wiki/parse/html.php
./wiki/parse/transforms.php
./wiki/parse/macros.php
./wiki/action/find.php
./wiki/action/view.php
./wiki/action/history.php
./wiki/action/diff.php
./wiki/template/common.php
./wiki/template/prefs.php
./wiki/template/find.php
./wiki/template/view.php
./wiki/template/diff.php
./wiki/templates/default/edit.xet
./wiki/templates/default/edit.xet
./wiki/template/history.php

Updated backup application:
./backup/dbset.php
./backup/egw_data_backup.php

Interface changes (images, templates, etc):
./wiki/templates/default/images/qstar.gif
./wiki/templates/default/images/nostar.gif
./wiki/templates/default/images/star.gif
./wiki/templates/default/images/desc.gif
./wiki/templates/default/images/asc.gif
./wiki/templates/default/images/neutral.gif
./wiki/templates/default/images/negative.gif
./wiki/templates/default/images/positive.gif
./forum/inc/hook_sidebox_menu.inc.php
./phpgwapi/templates/idots/footer.tpl
./phpgwapi/templates/idots/images/logonetmob.png
./phpgwapi/templates/idots/images/logonew.png
./phpgwapi/templates/idots/login.tpl
./phpgwapi/templates/idots/head.tpl
./phpgwapi/templates/idots/images/favicon.ico
./phpgwapi/templates/idots/navbar.inc.php
./phpgwapi/templates/idots/images/login-background.png
./phpgwapi/templates/idots/navbar.tpl
./phpgwapi/templates/idots/images/login-background.jpg
./phpgwapi/templates/default/images/favicon.ico
./phpgwapi/templates/default/head.tpl
./phpgwapi/templates/default/login.tpl
./phpgwapi/templates/prisma/login.tpl
./phpgwapi/templates/jerryr/login.tpl
./phpgwapi/templates/edge-it/login.tpl

Language files (new messages added, some errors fixed):
./wiki/setup/phpgw_en.lang
```

```
./wiki/setup/phpgw_it.lang
./registration/setup/phpgw_it.lang
./registration/setup/phpgw_en.lang
./phpgwapi/setup/phpgw_it.lang
./phpgwapi/setup/phpgw_en.lang
./forum/setup/phpgw_en.lang
./forum/setup/phpgw_it.lang

Installation-dependent settings (you will have your own version of this file!):
./header.inc.php
```

2.3 Storing reputation data in WikiRep

All reputation data are stored in the database. Currently, processing speed is an issue, because complete calculation of ratings and reputations for a page with few hundreds versions takes several second on a PC with ≈ 1 GHz processor. Therefore, the fields with pre-calculated reputation values have been added, to skip unnecessary calculations.

Users' opinions about page versions. Each page version accumulates *all* feedbacks it receives in fields `rating_sum`, `rating_sum2`, and `rating_n` in table `phpgw_wiki_pages`, as weighted sums of feedback marks, squares of marks, and number of marks. These three fields are floating-point numbers.

Additionally, each page version record in the `phpgw_wiki_pages` table has fields `rating_value` and `rating_quality`, that contain up-to-date *value* and *quality* of the global page version rating. These two fields are redundant, because they are calculated from the accumulated sums, but they are used to speed-up page visualization and processing and avoid re-calculations of the global page version ratings. Under assumption that the ratings are visualized more often than updated, this gives a performance improvement. These two fields are floating-point numbers.

Besides the accumulated values stored for each page version, there is a dedicated table `phpgw_wiki_pages_accounts` that contains records for specific pairs of user and page version. Currently, this table stores number of times each user has voted for each page version. Generally, it may store individual user-about-page opinions, which is not currently used. Each record in the table `phpgw_wiki_pages_accounts` corresponds to a unique pair of user and page version, and contains the following fields:

- `page_name`, `page_version` and `page_wiki_id` fields uniquely identify the target page version. *Wiki-ID* field is introduced for compatibility only, there is normally a single Wiki in the database, so the ID is always 0.
- `reviewer_id` is the identifier of the user.
- `review_mark` is the field that contains number of times the user has evaluated the page version.

This table may become very large, because the upper bound on its size is $(N_u \cdot N_{pv})$, where N_u is number of users and N_{pv} is total number of page versions in the Wiki.

Adoption coefficients and parent relations. Each page version record in the table `phpgw_wiki_pages` contains two fields to maintain the inter-version page structure: `parent_version` and `parent_adoption`. Field `parent_version` holds ID of the version (of the same page) that is identified as the parent for this version. The ID is an integer number. Field `parent_adoption` holds the adoption coefficient from the parent version to this version ($a_{parent\_version,this\_version}$). The field is a real number. If the version has no parent, the adoption coefficient is 0.0, and the parent ID is set to 0.

User-about-user opinions and users' reputations. The feedbacks also create digital opinions of users about each other. Each record in the table `phpgw_account_opinions` contains the following fields:

- `source_account_id` — ID number of the opinion's *originator* (subject).
- `target_account_id` — ID number of the evaluated user (opinion object).
- `opinion_sum`, `opinion_sum2` and `opinion_n` — raw aggregated values.
- `opinion_value` and `opinion_quality` — pre-calculated up-to-date values to speed-up processing.

3 Approaches to determine reputation quality

Key functions of WikiRep, rating inheritance and credit allocation, rely on pair-based representation of reputations and opinions. The opinions are processed as pairs $(value; quality)$, where *value* defines the 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. Both *value* and *quality* are derived from feedbacks generated by community members.

Deriving *value* is trivial in practice: mean or weighted average of evaluations is the most reasonable estimation. However, there is no unique way to define and calculate *quality*. The only necessary property of *quality* for WikiRep is that it is suitable as a weight for *value*. This requirement lacks a formal definition. In this Section we discuss the problem of choosing appropriate algorithm to derive reputation *quality* from a set of evaluations.

At the moment, we have implemented two approaches to define *quality*. The first one is a part of ROCQ framework [9, 10], hereafter referred to as Q-ROCQ. Q-ROCQ assumes that evaluations are real numbers sampled from a continuous distribution. The second approach is based on the assumption of discrete feedback distribution. We call it Q-PNE, Quality from Positive and Negative Evaluations. The motivation behind Q-PNE is that when feedback is generated by humans, it is preferable to use only a few fixed values. Most users prefer to choose from a smaller set of answers, and accuracy of fine-grained evaluation scale is depreciated by variability of evaluations among users. In the basic case, there are only two levels of evaluations: positive (corresponds to 1 mark) and negative (0 mark).

Below, we describe in detail both algorithms, and provide empirical comparison of them for the set of pre-constructed scenarios.

3.1 Q-ROCQ: Quality in ROCQ framework

Set of marks is treated as a random sample from an underlying continuous distribution. In the basic case, each mark is just a value without corresponding weight, in other words, all marks have weight of 1. The approach covers also the general case, when each mark has a corresponding quality weight. The set of evaluations is characterized by three values: (weighted) sum of evaluations *sum*, (weighted) sum of squares of evaluations *sum*2, and number of evaluations (or sum of weights) n. Pair (*value*; *quality*) is derived from feedback in the following way:

- sample mean (weighted average) becomes the value of the opinion pair: $value = sum/n$;
- *quality* is derived as probability that interval $[value - \Delta_r; value + \Delta_r]$ holds the actual mean of the underlying mark distribution. Width of the interval Δ_r is a system parameter to be chosen.

Quality characterizes accuracy of the estimated mean, as probability that the difference between actual and estimated means is less than Δ_r. The difference has *Student's t-distribution* [13]. The probability that absolute value of the difference is smaller than the chosen interval is given by the function:

$$A(t|n) = 1 - I_{\nu/(\nu+t^2)}\left(\frac{\nu}{2}, \frac{1}{2}\right) \ , \tag{1}$$

where $I_x(a, b)$ is incomplete beta function (11), $\nu = n - 1$ is the number of degrees of freedom, and t depends on width of the chosen interval:

$$t = \frac{\Delta_r \sqrt{n}}{S} \ , \tag{2}$$

where S is the unbiased standard deviation of the sample, given by:

$$S = \sqrt{\frac{sum2 - sum^2/n}{n-1}} \tag{3}$$

Overall, *quality* is found as:

$$quality = 1 - I_{(n-1)/(n-1+t^2)}\left(\frac{n-1}{2}, \frac{1}{2}\right) \ , \text{ where} \tag{4}$$

$$t^2 = \frac{\Delta_r^2 n^2 (n-1)}{n \cdot sum2 - sum^2} \tag{5}$$

More details on calculation of $I_x(a, b)$ and source code that was adopted in the implementation are found in [13]. The algorithm above cannot be used for $n \leq 1$.

Quality has a dual role: it is defined as probability that actual mean is close enough to the estimated *value*, and is used as weight for the corresponding *value*. This substitution complies with the requirements imposed on the *quality*.

3.2 Q-PNE: Quality from Positive and Negative Evaluations

The only difference in initial assumptions between Q-PNE and Q-ROCQ is that with Q-PNE treats the collected feedbacks as if they are sampled from i.i.d. Bernoulli distribution with probability of a positive evaluation p_{actual}. In other words, an evaluation for a given object is either positive with probability p_{actual}, or negative with probability $(1-p_{actual})$, independently from other evaluations. Probability p_{actual} characterizes merit of the object and is not known. In this case, *value* of reputation is given by the estimation of p_{actual}. The change of assumptions leads to a different way to calculate *quality*.

Let n be the number of collected feedbacks, and k be the number of positive ones among them (for now, we assume that all feedbacks have weight 1). Then estimation of p_{actual} (i.e. *value*) is given by $\nu = k/n$. We define *quality* of the opinion as probability that the hypothesis H_0 below is not rejected for

$$p_{lo} = max(\nu - \Delta_r; 0) \tag{6}$$

$$p_{hi} = min(\nu + \Delta_r; 1) \tag{7}$$

Hypothesis H_0*:* Given the sample of collected feedbacks, probability p_{actual} lies in the interval $[p_{lo}, p_{hi}]$.
In other words, *quality* is the confidence level for the statistical test of the hypothesis H_0.

Probability that for $p_{actual} = p$ exactly k of n feedbacks are positive is given by the binomial distribution:

$$p_{binomial}(n, k, p) = C(n, k) p^k (1-p)^{n-k} \tag{8}$$

$C(n, k)$ is defined using gamma-function $\Gamma(x)$, when n and k are not integer. By this, we relax the assumption that all weights are equal to 1.

Because probability p_{actual} is a continuous variable, we define probability density function for it, conditioned to the observation that k out of n feedbacks are positive:

$$f_{n,k}(p) = P(p = p_{actual} | k \text{ of } n \text{ sample are positive }) = \frac{C(n,k)p^k(1-p)^{n-k}}{\int_0^1 C(n,k)p^k(1-p)^{n-k}dp} , \tag{9}$$

Then, the maximal confidence level to not reject the hypothesis (i.e. *quality*) is:

$$q = \frac{\int_{p_{lo}}^{p_{hi}} C(n,k)q^k(1-q)^{n-k}dq}{\int_0^1 C(n,k)q^k(1-q)^{n-k}dq} , \tag{10}$$

The integrals are expressed using beta function $B(a,b)$ and incomplete beta function $I_x(a,b)$:

$$B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1}dt \tag{11}$$

$$I_x(a,b) = \frac{1}{B(a,b)} \int_0^x t^{a-1}(1-t)^{b-1}dt \tag{12}$$

Finally, transformations of (10) give expression for *quality*:

$$quality = I_{\nu+\Delta_r}(n+1, n-k+1) - I_{\nu-\Delta_r}(n+1, n-k+1) \tag{13}$$

3.3 Empirical comparison of Q-ROCQ and Q-PNA

In this Section we observe dynamics of quality evolution in several constructed scenarios. Each feedback takes one of three values: positive, neutral or negative, corresponding to numerical values 1.0, 0.5 and 0.0. Weight of each each feedback is 1.0. Value of Δ_r in both algorithms is chosen in accordance with the reputation visualization scheme described in Section 5. In the current implementation, the visualization provides 6 levels for values from $[0,1]$ range, giving resolution of 0.167. Therefore, we use for both algorithms $\Delta_r = 0.08$, corresponding to width 0.16 of confidence interval, close to the resolution of the rating representation. For $n \leq 1$ Q-ROCQ returns default $quality = \Delta_r = 0.08$. The scenarios considered are:

- **all-positive:** all incoming feedbacks are positive;
- **uniform:** one third of feedbacks are positive, one third neutral and one third negative;
- **all-neutral:** all feedbacks are neutral;
- **confrontation:** half of the feedbacks are negative and the other half are positive;
- **majority (99%):** 99% of feedbacks are positive and 1% negative;
- **majority (80%):** 80% of feedbacks are positive and 20% negative;

For symmetric cases with negative and positive feedbacks being swaped, *quality* behaves exactly the same. For example, a case of **all-negative** will be equal to **all-positive**.

From the plots on Figure 1, several observations follow:

- for all-positive and all-neutral scenarios, and (less sharply) for majority-99%, Q-ROCQ saturates to maximum $quality = 1.0$. This happens because of low variation in input data. The effect makes the reputation scheme vulnerable to collusion;
- in scenarios, when there is substantial diversity in the input data (uniform, confrontation and majority-80%), both algorithms exhibit similar behavior (difference in *quality* is less than 0.1);

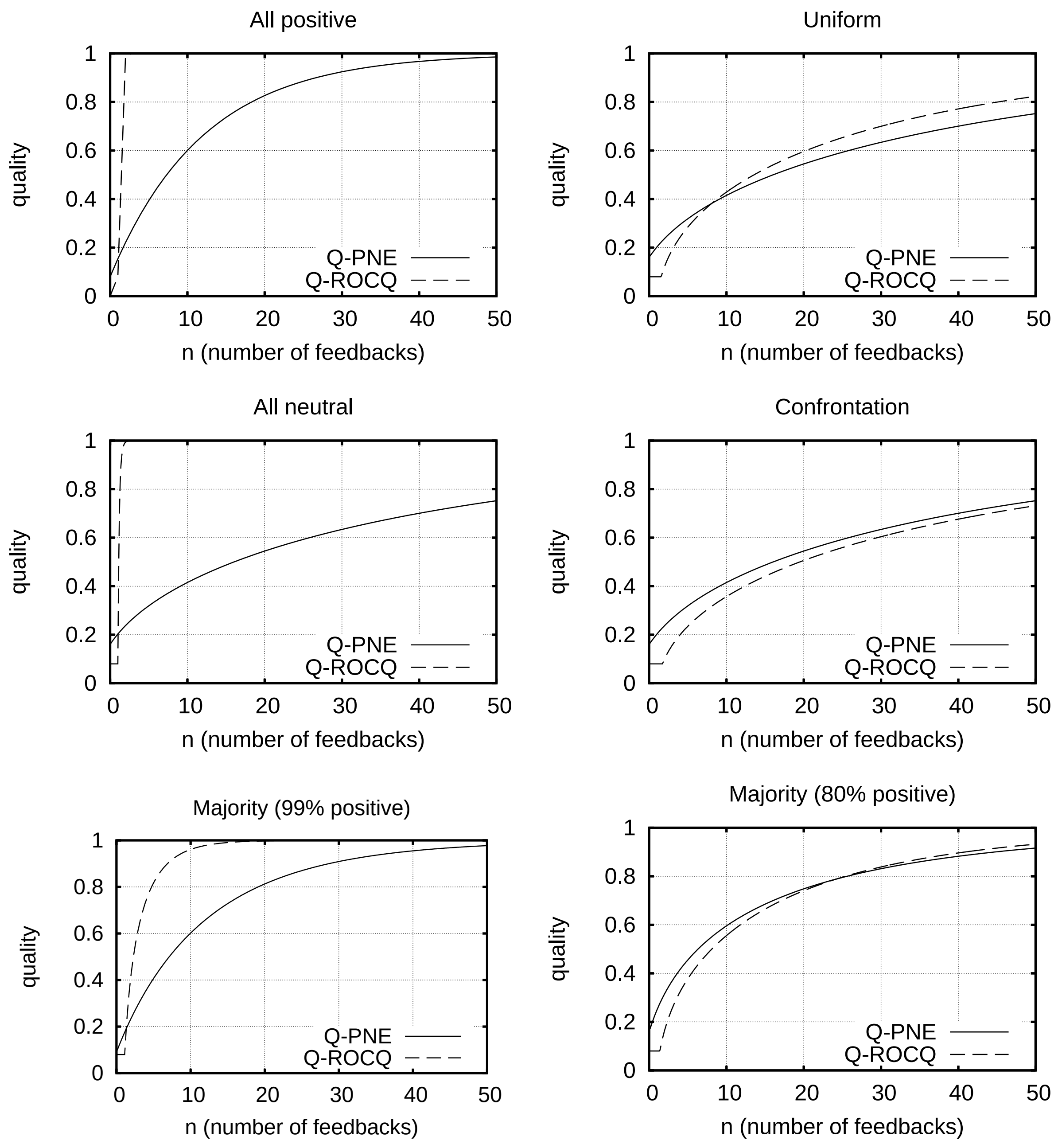

Figure 1: Quality dynamics for different scenarios

- Q-PNE does not distinguish confrontation and all-neutral cases, thus confusing mediocre and questionable reputations;
- *quality* provided by Q-PNE in all-neutral case is less than in all-positive case.

All the observations above follow from the assumptions of the algorithms: Q-ROCQ does not expect that evaluations have exactly the same value, and Q-PNE is not aware of neutral evaluations. We conclude that both of the algorithms have weak points in the given setting, and further improvement is necessary.

4 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. An adoption coefficient $a_{i,j}$ determines how much content from previous version i is preserved untouched in newer version j. 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 a metric similar to *edit distance* between the texts is found, using the blocks as characters [12]. The distance used is the minimal number of insert and delete operations on characters, needed to transform one text into the other: $(N_{inserted} + N_{deleted})$. 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:

$$a_{old,new} = 1 - \frac{N_{inserted} + N_{deleted}}{N_{total,new} + N_{deleted}}, \tag{14}$$

Property: *The adoption coefficients as defined above are symmetric:*

$$a_{i,j} = a_{j,i}, \text{ for all } i, j \tag{15}$$

Proof. Let us define as $N^{i\to j}_{inserted}$ and $N^{i\to j}_{deleted}$ results of comparison that considers version i as *old* and j as *new*, accordingly to (14). Correspondingly, $N^{j\to i}_{inserted}$ and $N^{j\to i}_{deleted}$ refer to comparison of version j as *old* to version i as *new*. N^{i}_{total} and N^{j}_{total} are the total number of blocks in the versions.

The following equations define the relations between total number of blocks in the versions:

$$\begin{array}{c} N^{i}_{total} + N^{i\to j}_{inserted} - N^{i\to j}_{deleted} = N^{j}_{total} \\ N^{i}_{total} = N^{j}_{total} + N^{j\to i}_{inserted} - N^{j\to i}_{deleted} \end{array} \tag{16}$$

Because $(N_{inserted} + N_{deleted})$ is minimized by the comparison algorithm, (17) holds. Otherwise the greater sum can be minimized to the value of the lesser one.

$$N^{i\to j}_{inserted} + N^{i\to j}_{deleted} = N^{j\to i}_{inserted} + N^{j\to i}_{deleted} \tag{17}$$

So, nominator of the fraction in (14) is the same for $a_{i,j}$ and $a_{j,i}$.

From equations (16) and (17) immediately follows that

$$\begin{array}{c} N^{i\to j}_{inserted} = N^{j\to i}_{deleted} \\ N^{j\to i}_{inserted} = N^{i\to j}_{deleted} \end{array} \tag{18}$$

Finally, (16) and (18) give:

$$N^{i}_{total} + N^{j\to i}_{deleted} = N^{j}_{total} + N^{j\to i}_{inserted} = N^{j}_{total} + N^{i\to j}_{deleted} \tag{19}$$

and denominator of the fraction in (14) is also the same for $a_{i,j}$ and $a_{j,i}$, and thus $a_{i,j} = a_{j,i}$. $\square$

5 Visualizing ratings and reputations

As a simple and intuitive way to visualize reputations and ratings we have chosen ‘stars’, an approach similar to that of *epinions.com*, *amazon.com reviews*, and others. The value of rating is translated into integer number of stars, from 0 to 5. If we do not consider *quality*, number of stars corresponding to a *value* from $[0, 1]$ range is found as

$$N_{stars} = \lfloor 5.99 \cdot value \rfloor \ , \tag{20}$$

which gives 6 possible values from 0 to 5, depicted in Table 1.

Table 1: Reputation representation using stars (*quality* not considered)

value:	[0, 0.17)	[0.17, 0.34)	[0.34, 0.5)	[0.5, 0.67)	[0.67, 0.84)	[0.84, 1.0]
N_{stars}:	☆☆☆☆☆	★☆☆☆☆	★★☆☆☆	★★★☆☆	★★★★☆	★★★★★

The expression (20) only considers *value* of the reputation, which is not feasible in our system. Visualized reputation has to reflect *quality* of the reputation as well. We achieve this by using two types of stars: *full* and *partial*, reflecting the **range** of reputation values, that takes into account *quality*. For the lower bound, we assume that the unknown part of the reputation has $value = 0.0$. For the upper bound, the unknown part of the reputation is considered to have $value = 1.0$. The unknown part has the weight $(1 - quality)$, because *quality* is used as a reliability weight.

$$\begin{aligned} value\_lower\_bound &= value \cdot quality \\ value\_upper\_bound &= value \cdot quality + (1 - quality) \end{aligned} \tag{21}$$

Number of stars of both types is then:

$$\begin{aligned} N_{stars\_full} &= \lfloor 5.99 \cdot value\_lower\_bound \rfloor \\ N_{stars\_part} &= \lfloor 5.99 \cdot value\_upper\_bound \rfloor - N_{stars\_full} \end{aligned} \tag{22}$$

Table 2 list several examples of resulting reputations for several combinations of *value* and *quality*, and Figure 2 presents a plot of correspondence between reputations and ‘star’ visualization. If *quality* is low, many *partial* stars are present, and as quality increases, more *full* or *empty* stars appear instead, depending on the reputation value.

Table 2: Examples of reputation representation using *quality*

N	*value*	*quality*	*value_lower_bound*	*value_upper_bound*	stars (full/partial)
1	1.0	0.84	0.84	1.0	★★★★★ (5/0)
2	0.99	0.82	0.81	1.0	★★★★⯪ (4/1)
3	0.92	0.67	0.62	1.0	★★★⯪⯪ (3/2)
4	0.66	0.84	0.55	0.71	★★★⯪☆ (3/1)
5	0.48	0.72	0.34	0.62	★★⯪☆☆ (2/1)
6	0	0.08	0	0.92	⯪⯪⯪⯪⯪ (0/5)
7	0.16	0.67	0.11	0.44	⯪⯪☆☆☆ (0/2)

References

[1] eGroupWare application development. http://cvs.sourceforge.net/viewcvs.py/egroupware/phpgwapi/doc/Attic/index.txt?rev=1.4, retrieved on September 2005.

[2] eGroupWare portal. http://www.egroupware.org/, retrieved on September 2005.

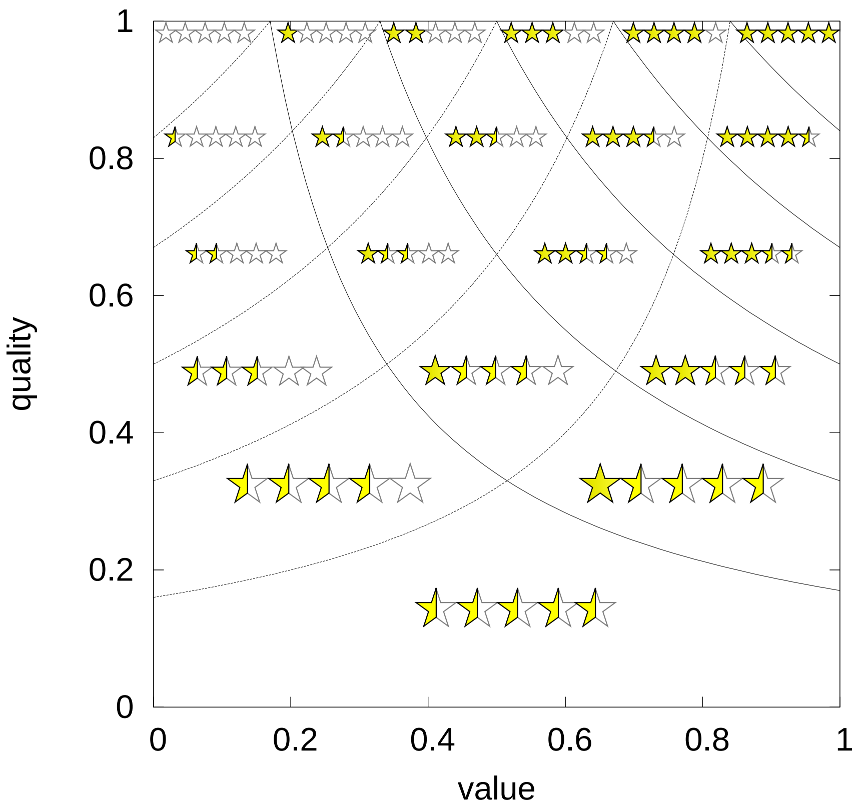

Figure 2: Visualization of reputations and ratings by stars

[3] eTemplate — templates and dialog-editor for eGroupWare. http://www.egroupware.org/egroupware/etemplate/doc/etemplate.html, retrieved on September 2005.

[4] eTemplate: widget based template system for eGroupWare. http://www.egroupware.org/index.php?page_name=&category_id=38&wikipage=eTemplate, retrieved on September 2005.

[5] Ralf Becker. Documentations for developers from eGroupWare: Coding standards. http://www.egroupware.org/index.php?page_name=&category_id=38&wikipage=CodingStandards, retrieved on September 2005.

[6] Ralf Becker and eGroupWare developers. Documentations for developers from eGroupWare: Style guide. http://www.egroupware.org/index.php?page_name=&category_id=38&wikipage=StyleGuide, retrieved on September 2005.

[7] eGroupWare developers. Requirements to install eGroupWare. http://www.egroupware.org/requirements, retrieved on December 2005.

[8] Joshua Eichorn and Greg Beaver. phpDocumentor. http://www.phpdoc.org/, retrieved on September 2005.

[9] 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.

[10] 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.

[11] Reiner Jung. How to install and secure eGroupWare. http://www.csit.fsu.edu/twiki/pub/TWiki/FileAttachment/How_to_install_and_secure_eGroupWare_05.pdf, 2005.

[12] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[13] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C, 2nd. edition.* Cambridge University Press, 1992.

[14] Mikalai Sabel. WikiRep: phpdocs. http://dit.unitn.it/ msabel/wikirep-phpdoc/, 2005.

[15] Mikalai Sabel, Anurag Garg, and Roberto Battiti. Wikirep: Digital reputation in virtual communities. Technical Report DIT-05-050, Dipartimento di Informatica e Telecomunicazioni, Università degli Studi di Trento, Trento, Italy, June 2005.