

# GRAPPLE

D6.2b Version: 1.0

## Integrated user model rating and reputation system

|                               |   |
|-------------------------------|---|
| <b>Document Type</b>          | Deliverable   |
| <b>Editor(s):</b>             | Dominikus Heckmann  |
| <b>Author(s):</b>             | Dominikus Heckmann, Rafael Math, Fabian Abel, Eelco Herder, Jan Hidders |
| <b>Reviewer(s):</b>           | Gordon Power (TCD), Kees van der Sluijs (TUE)                           |
| <b>Work Package:</b>          | WP6   |
| <b>Due Date:</b>              | 01-08-2010  |
| <b>Version:</b>               | 1.0   |
| <b>Version Date:</b>          | 31-01-2011  |
| <b>Total number of pages:</b> | 24  |

**Abstract:** This document describes the design and the implementation of an integrated user model rating and reputation system. It is based on the higher goal to reach the level of “trust” between the different participating LMSs or “trust in the World Wide Web” in general. An overview over the scientific background is presented and design decisions that have been realised in the implementation phase will be discussed . This deliverable D6.2b also contains the deliverable D6.2a.

**Keyword list:** Semantic Web, Rating, Reputation, User Modelling, Web 2.0

## Summary

In a distributed user modelling scenario the shift to Web 2.0 leads to the idea that the quality, the trustworthiness and the correctness of statements about users can be judged by the so called “wisdom of the crowds”. In a distributed user model scenario, the quality of the user model data can vary and make decisions based on user models especially difficult. The aim of this deliverable is to enrich the user models with quality ratings to support the trust-idea of the Semantic Web approach as well as a social computing approach to improve the user models. The design and implementation of a rating system for Grapple Statements will be presented as well as the design and implementation of a reputation system for information providers.

## Authors

| Person             | Email                           | Partner code |
|--------------------|---------------------------------|--------------|
| Dominikus Heckmann | heckmann@dfki.de                | DFKI         |
| Jan Hidders        | j.hidders@tudelft.nl            | TUD          |
| Rafael Math        | Rafael.Math@dfki.de             | DFKI         |
| Fabian Abel        | abel@l3s.de & f.abel@tudelft.nl | L3S & TUD    |
| Eelco Herder       | herder@l3s.de                   | L3S          |

## Table of Contents

|   |           |
|---|-----------|
| <b>SUMMARY</b> .....                              | <b>2</b>  |
| <b>AUTHORS</b> .....                              | <b>2</b>  |
| <b>TABLE OF CONTENTS</b> .....                    | <b>2</b>  |
| <b>TABLES AND FIGURES</b> .....                   | <b>3</b>  |
| <b>LIST OF ACRONYMS AND ABBREVIATIONS</b> .....   | <b>4</b>  |
| <b>1 INTRODUCTION</b> .....                       | <b>5</b>  |
| <b>2 RELATED WORK</b> .....                       | <b>5</b>  |
| <b>2.1 Collective Election Systems</b> .....      | <b>6</b>  |
| <b>2.2 Rating Systems</b> .....                   | <b>7</b>  |
| <b>2.3 Trust &amp; Reputation</b> .....           | <b>8</b>  |
| <b>3 DESIGN</b> .....                             | <b>9</b>  |
| <b>3.1 Design of the Rating System</b> .....      | <b>9</b>  |
| 3.1.1 What can be rated in GRAPPLE? .....         | 9         |
| 3.1.2 Who is allowed to rate in GRAPPLE? .....    | 10        |
| 3.1.3 Initial Design of the Database Schema ..... | 10        |
| 3.1.4 Initial Design of the User Interface .....  | 11        |
| <b>3.2 Design of the Reputation System</b> .....  | <b>11</b> |
| 3.2.1 Design of the Trust Model .....             | 11        |

|            |   |                                     |
|------------|---|-------------------------------------|
| 3.2.2      | Design of the Database Schema.....                  | 11                                  |
| 3.2.3      | Design of the User Interface.....                   | 12                                  |
| <b>4</b>   | <b>IMPLEMENTATION .....</b>                         | <b>12</b>                           |
| <b>4.1</b> | <b>Implementation Details: Rating .....</b>         | <b>12</b>                           |
| 4.1.1      | Rating Documentation .....                          | 14                                  |
| <b>4.2</b> | <b>Implementation Details: Reputation .....</b>     | <b>15</b>                           |
| 4.2.1      | Reputation Documentation .....                      | 16                                  |
| <b>4.3</b> | <b>Implementation Details: Integration.....</b>     | <b>17</b>                           |
| 4.3.1      | Integration into GRAPPLE Event Bus Technology.....  | 17                                  |
| 4.3.2      | Integration into GRAPPLE User Model Framework ..... | 17                                  |
| <b>5</b>   | <b>DISCUSSION &amp; SUMMARY .....</b>               | <b>18</b>                           |
| <b>6</b>   | <b>APPENDIX: SELECTED SOURCE CODE EXAMPLES.....</b> | <b>19</b>                           |
| 6.1.1      | Xajax Functions .....                               | 19                                  |
| 6.1.2      | Statement Rating Functions .....                    | 19                                  |
| 6.1.3      | Reputation Functions.....                           | 22                                  |
|            | <b>REFERENCES .....</b>                             | <b>23</b>                           |
|            | <b>DOCUMENT CONTROL .....</b>                       | <b>ERROR! BOOKMARK NOT DEFINED.</b> |

## Tables and Figures

### List of Figures

|  |    |
|--|----|
| Figure 1: Trust as top of the Semantic Web Layer Cake by Tim Berners-Lee .....                                 | 5  |
| Figure 2: Instantiated Database Schema for Testing Basic Ratings .....   | 10 |
| Figure 3: User interface widget for a 5-scale rating .....   | 11 |
| Figure 4: Multi-dimension ration (in analogy to [4]).....  | 11 |
| Figure 5: Reputation-Matrix: the active user adds reputation medals to information providers .....             | 12 |
| Figure 6: Data flow chart of the rating system .....   | 13 |
| Figure 7: User “dominik” submitted the rating “very good” (4) for statement “S..AB9C72-3JFC8I”.                | 13 |
| Figure 8: Statement “S..AB9C72-3JFC8I” has been rated once and has the average rating of 4.013                 |    |
| Figure 9: Screenshot of rated GrappleStatements .....  | 15 |
| Figure 10: Data flow chart of the reputation system .....  | 15 |
| Figure 11: Example of database schema entries.....   | 16 |
| Figure 12: Screenshot Reputation-Matrix: the active user adds reputation medals to information providers ..... | 16 |
| Figure 13: Event Listener via GUMF WP2 & WP6 module .....  | 17 |
| Figure 14: Client Application of the GUMF Architecture .....   | 18 |

## List of Acronyms and Abbreviations

|                  |  |
|------------------|--|
| GALE             | GRAPPLE Adaptive Learning Environment  |
| GRAPPLE          | Generic Responsive Adaptive Personalised Learning Environment                        |
| GEB              | GRAPPLE Event Bus  |
| GrappleBroker    | A service to store and retrieve partial user models represented as GrappleStatements |
| GrappleStatement | A Unit of information about a user together with contextual meta data.               |
| GUMF             | GRAPPLE User Model Framework   |
| LMS              | Learning Management System   |
| OWL              | Ontology Web Language  |
| RDF              | Resource Description Framework   |
| TEL              | Technology-Enhanced Learning   |
| UM               | User Model (or sometimes "User Modelling") a set of GrappleStatements                |
| WP               | Work Package   |

# 1 Introduction

In a distributed user modelling scenario that is based on the Semantic Web, the shift to Web 2.0 leads to the idea that the quality, the trustworthiness and the correctness of statements about users can be judged by the so called “wisdom of the crowds”. In a distributed user model scenario, the quality of the user model data can vary and make decisions based on user models especially difficult. The term “distributed” is used here in the sense of heterogeneous and inhomogeneous data from different information providers that have not harmonised their models. The aim of this deliverable is to enrich the user models with quality ratings to support the trust-idea<sup>1</sup> of the Semantic Web approach as well as a social computing approach to improve the user models. The integration into the rest of the GRAPPLE system is simply realised by including the ratings into the basic information unit; the GrappleStatements.

This deliverable consists of a design of extensions to the D6.1a/b services, taking into account that different (parts of) distributed user models must be rated differently, which leads to ranking the reliability of user model values. The design chapters are followed by an implementation chapter. This deliverable D6.2b) builds on the a) deliverable.

The overall idea of this deliverable is to enrich the user models with quality ratings to support the trust-idea of the Semantic Web approach (see Figure 1) as well as a social computing approach to improve the user models.

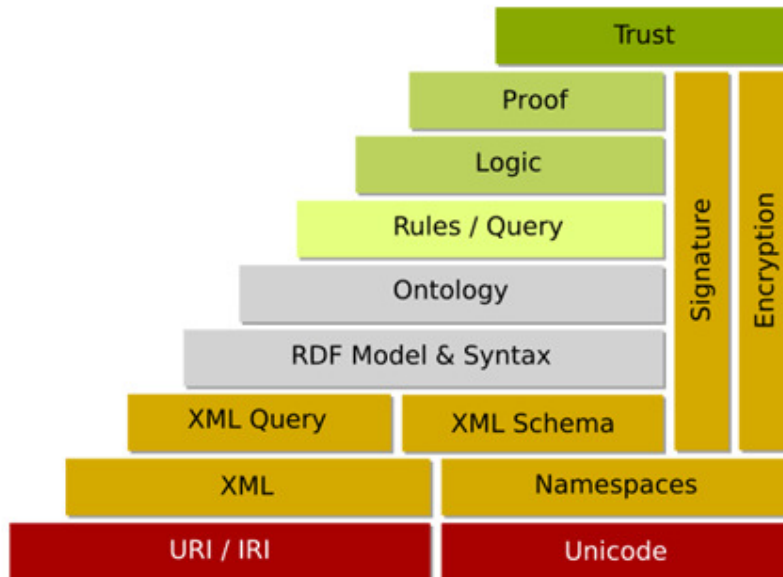


Figure 1: Trust as top of the Semantic Web Layer Cake by Tim Berners-Lee

# 2 Related Work

This section briefly discusses several approaches that are related to rating, trust and reputation systems on the Web.

In general, a rating is the *subjective* evaluation or assessment of something, in terms of quality, quantity, or some combination of both. Most rating systems centre around rating content, often user-contributed content and they frequently help apply community values and acclaim to that content. Users can rate a service by

<sup>1</sup> At the time Tim Berners-Lee defined the Semantic Web layer cake, the notion of “trust” is built on logic and proof, on signature and encryption. This pure semantic-web-notion of trust is extended into a social-semantic-web-notion of trust that also builds on the “wisdom of the crowds”.

using positive/neutral/negative values (e.g. eBay rating system), 5-point scales (e.g. using stars), or 10-point scales.

In [5], there are three important elements of a good rating system: granularity, distinction and a statistical basis. Granularity means that the users have more options such that the users have a greater ability to differentiate their ratings. For example, a 5-scale rating that allows half-points (e.g. 0.5, 1.0, 1.5, ..., 5.0) gives the user more options for expressing their ratings compared to the one without half-points (e.g. 1, 2, 3, 4, and 5). A problem that could occur when a user gives ratings, is inconsistency in his/her ratings over time. If a user rates everything low for a while, and then rates everything high, then he/she has a bigger chance of biasing the overall rating pool. Worse, his/her individual ratings aren't meaningful, because, for example, we cannot look at two items, see that one is a "2" and another is a "4", and truly believe that the user likes the "4" a fair amount more than the "2". Therefore, the criteria of the ratings have to be distinct in order to help the users to stay consistent and give more meaningful ratings. The following is an example of distinct rating systems with 5-scale rating for a game.

- 5 – Excellent game. Always want to play.
- 4 – Good game. I like to play.
- 3 – Average game, slightly boring, take it or leave it.
- 2 – Bad game, likely won't play this again although could be convinced.
- 1 – Extremely annoying game, won't play this ever again

Finally, a good rating system should be statistically sound. This means that the average of all given ratings of a certain item should be meaningful to the users. Of course, the best way to make the ratings statistically sound is with volume. By having, for example, thousands of ratings for an item, any anomalies will become noise. One way to compute the average ratings that are statistically sound (apart from normal average ratings) is by applying the Bayesian average. The idea behind a Bayesian average is to normalise ratings by pushing them toward the average rating for an item.

## 2.1 Collective Election Systems

Collaborative rating - seen under the social Web 2.0 community approach - points towards collective election systems. The long history of collective election systems continues today in the World Wide Web. According to Allen and Appelcline [6], especially people of the western culture appreciate a high participation in election processes. The reason for this syndrome is the belief that the fairest result can be achieved if a great diversity of people takes part in the election process.

In the course of time, three distinct kinds of election systems evolved, which differ in the intended results: subjects should either make a choice between objects, give their opinion about objects or compare different objects.

So-called selection systems allow the target-oriented selection between several elements. Selection systems include representative systems and consultative systems, both of which are political election systems. In representative systems, the election is performed through a delegate (representative) like in the republican election system. Consultative systems constitute the real idea of democracy because the election is done by direct ballot with a decision by majority. A variant of the consultative systems is formed by the consensus systems (e.g. the Westminster system), in which the power of the masses is stronger than the one of an individual; thus- the "tyranny of the majority" can be avoided.

A subcategory of the selective systems is built by opinion systems, which mainly act as an indicator of how people have decided in an election or how they will decide in the near future. These systems constitute surveys before an election or first projections during an election.

The third type of election systems is comparison systems, which support the measurement of individual elements among themselves. They can be divided into three categories: comparing ranking systems, which are mainly objective and classify participants of a competition; comparing rating systems, which mix up objective and subjective opinions and are used to especially rate entities (e.g. the product rating system of Amazon.com); reputation systems mix up objective and subjective opinions as well, but are rather used to rate people than things.

## 2.2 Rating Systems

This section describes the concept of rating systems in general as well as different criteria that are essential for a good rating system using eBay's rating system as an example.

The online auction house eBay was one of the first web sites to make use of rating systems in an extensive way. Their rating system was considered as an innovation in the field of e-commerce and should be used to determine the trustworthiness of the users by providing facilities to give a positive, negative, or neutral feedback for each transaction. The feedback score is calculated as the number of unique positive feedback minus the number of unique negative feedback. Unique feedback means that repeated ratings performed by the same user in the same week are excluded. The positive feedback score is calculated by dividing the number of positive feedback ratings by the total number of feedback ratings obtained, resulting in a percentage value.

Additionally, the user can add an arbitrary textual description of the transaction rating.

As almost all ratings given by eBay users are positive however, the feedback scores have a very large expressiveness because they do not tell much more than how many transactions have been performed by the respective member.

Allen and Appelcline investigated several rating systems [6], among others also the one of eBay. They defined the following criteria that are essential for the quality of a rating system:

- A rating system has to be granular. Statistics show that people tend to ignore the lower part of the scale of rating systems. Consequently, the decimal places of mean feedback scores are of decisive importance. The rounding of mean scores or the simplification of ratings to a "thumbs up" or "thumbs down" would be the wrong strategy.
- One major issue with rating systems is the inconsistent ratings performed by the user. If a user expresses the same opinion in two different rating processes with two distinct ratings, the feedback statistic is influenced. That's why the user should be supported to remain consistent by creating clearly defined meaning which feedback value corresponds to which opinion of the object. The larger the rating scale is the larger is the risk that the user gets confused.
- Rating systems must be statistically reliable. The best way to realise this is to collect a large number of ratings. If there are only a few unique ratings for an object, the risk of an unreliable overall rating increases. The more ratings are performed for an object, the smaller the effect of outliers is. Optimally, one should assign a smaller significance to objects with very few ratings.
- If users rate each other, there is a high risk that there are almost only positive ratings. The reason for this effect is that users are afraid of getting a negative rating in revenge for a negative feedback. That's why a rating system should not be bilateral.
- User ratings should have a concrete usage within a web site. A possible way to achieve this is to display rankings, in which the best and the worst rated objects can be investigated.
- A clear user interface is essential for a rating system as badly designed, confusing graphical user interfaces discourage the user; and have a negative effect on the quality of the ratings..

Although being part of eBay's success story, Allen and Appelcline pointed out that the rating system of eBay in its early version shows the following weaknesses:

- It is not granular because it only supports the three choices of positive, neutral and negative feedback.
- There are no clear guidelines which user behaviour should result in which rating.
- The rating system is statistically not reliable. The rating only depicts a rough amount of performed transactions rather than a real, subjective measure for the trustworthiness of the user.
- As the users rate each other, they hesitate to give negative feedback because they are afraid of getting a negative rating in revenge.

- The rating system basically does not have significance because there is no possibility to exclude users with a low rating score from auctions, for instance.

In May 2007, eBay introduced a reworked version of the rating system [11]. The so-called detailed rating makes it possible to rate additional aspects of a transaction rather than only providing the three categories positive, neutral and negative. Using the following questions, the transparency of the auction platform shall be increased:

1. Was the item delivered as described?
2. How would you rate the communication with the seller?
3. How long took the shipping time?
4. Were the shipping and handling charges adequate?

Using the new five-star rating system, users can give detailed statements within these additional rating dimensions. The mean value of these rating categories is shown in the user profile, among the standard feedback score.

### 2.3 Trust & Reputation

In the real world, a group of users provides ratings for items (in our case also the information providers, apart from the GrappleStatements, see Section 3.1). However, not all users who rate the information provider are trustworthy. Hence, the reputation of this group of users has to be determined. Note that not all given ratings are fair, reliable and trustworthy. Reputation is the social evaluation of the public towards a person, a group of people, an organisation or an information source. Reputation systems are often useful in large online communities in which users may frequently have the opportunity to interact with users with whom they have no prior experience or in communities where user generated content is posted. In many cases, reputation is used to establish trust. The past interactions or performance of an information source captured in its reputation are combined to assess its future behaviour. Once the reputation is determined, trust can be established.

Zhang and Cohen [18] proposed a method to determine the trustworthiness of *advisors* - users who provide ratings for the information provider. Note that the existence of malicious advisors is discussed in [19]. This method determines the trustworthiness of advisors by computing and combining two values: *private* and *public reputation values*. The private reputation of an advisor is based on ratings that are given by the advisors to information providers with whom the user has already had some experience. The intuition is that if the advisor is reputable and has similar preferences as the user, then the ratings given by the user and advisor are likely similar. The public reputation value is important in the case of lacking private knowledge about the advisor. This reputation value is based on the advisor's ratings of all information providers in the system. The combination of the weighted private and public reputation values can be used to represent the trustworthiness of the advisor. The weight is significant in specifying the reliability of reputation values. For example, the user who relies on private reputation value will give more weight to private reputation value and will less consider the public reputation values.

In [19], Richardson et al. also addressed the problem of possibly unfair or unreliable ratings. The approach in [19] provides a means of merging trust that is robust to noise and places emphasis on personalised trust. One interesting feature of this approach is the propagation of trust through a users' network. Note that this approach is described as a generalisation of PageRank [20] to the Semantic Web. In this approach, a user is required to build his/her Web of Trust, a small set of users whom he/she trusts. Based on this Web of Trust, the trust values of all other users are composed. The trust value of an information provider is then derived and computed using some aggregation functions along each possible chain of trust from the user to the information provider. The result of this computation does not agglomerate "trustworthiness" of each user. Instead, each user receives a personalised set of trusts, which may vary widely from person to person.

Additionally, the domain of knowledge (context) can be considered while computing the trust of information provider. Ding et al. [21] addressed this issue and elaborated on several types of trust: *domain expert trust* (trust in an agent's domain knowledge), *recommendation expert trust* (trust in an agent's ability to refer other agents), *similar trusting trust* (two agents having similar trust in other agents), and *similar cited trust* (two agents being similarly trusted by other agents). This context consideration is also supported by Bizer and Oldakowski [22]. In [22], Bizer and Oldakowski made several claims: (1) any statements contained in the Semantic Web must be considered as claims rather than facts until trust can be established, (2) it is too much of a burden to provide trust information that is current, (3) context-based trust matters and (4) it is possible to use "content-based trust" using common sense rules of the world to make a trust decision.

## 3 Design

### 3.1 Design of the Rating System

After the literature survey in the related work, it has been decided to introduce a distinct five-scale rating model because the most prominent and most effective social web pages and related work implement such a five-scale rating. The path of the wisdom of the crowds has been taken. Now a closer look at questions that meet the ideas of the GRAPPLE scenarios needs to be taken.

#### 3.1.1 What can be rated in GRAPPLE?

First of all, the basic unit of the GRAPPLE user modelling framework GUMF is subject to be rated: the so called GrappleStatement, as defined and introduced in deliverable D6.1a:

**Definition 1:** *A GrappleStatement is defined as an information unit, enriched with meta-data about it.*

The final syntactical representations of GrappleStatements are defined in deliverable D2.1 and described in D7.2 in a Semantic Web language. For this deliverable D6.2a), it is only of importance that each GrappleStatement can be treated as a resource, which means each statement can be referenced by a unique identifier.

Examples of such GrappleStatements in pure textual form are:

S1 := Mary likes chemistry

S4 := Mary prefers learning style A to learning style B

S7 := Mary is colour-blind

S8 := Peter likes mathematics

S9 := Peter is a good Java programmer (claimed by Mary)

S10 := Peter knows the concept of object oriented programming (learned with the AHA! system)

S12 := Peter knows the Theorem of Pythagoras (learned with Moodle, last year)

S13 := Mary does not like chemistry very much (claimed by Peter, yesterday)

These examples as shown above refer to pure statements (S1-S8) and statements with metadata information (S9-S13). An interesting conflict holds between S1 and S13: both claim opposite values about Mary's interest in chemistry. The whole motivation of this work package task D6.2 is to allow everybody, or selected people, to "rate" statements, in order to compare them and in order find out the truth, or better in terms of Web 2.0: the wisdom of the crowds.

Now the question arises, what part of the statement should be rated? Since the GrappleStatements consist of a main part and a meta part, both can be the subject of the rating. Interesting dimensions are for example:

- Actuality of the (main part) information (with a temporal dimension)
- Correctness of the (main part) information
- Trustworthiness of the information (influenced by meta parts like "creator of the statement")

The second subject that could be rated directly – apart from GrappleStatements – is the creator of the statements, the so-called information provider directly. This will be discussed in Chapter 4 under the topic of "Trust and Reputation".

### 3.1.2 Who is allowed to rate in GRAPPLE?

According to [23], peer production is gaining acceptance for the creation of structured knowledge and semantic metadata. Peer production refers to a web-based production model where a large number of contributors work on a common project.

In this deliverable, rating-based incentive mechanisms are proposed to assure the quality of structured knowledge created collaboratively. Users review contributions created by other users and LMSs and rate them according to the quality perceived. To motivate users to contribute to the community, they are rewarded with points according to the extent and quality of their contributions. In turn the quality of contributions is computed, based on their ratings.

The weight of the rating can be adjusted according to who is rating. This leads to the question:

Who should be allowed to rate statements in GRAPPLE and the information provider? Possible candidates are for example:

- The creator (can be everybody)
- The described person (user/student)
- The peer of the described person
- The teacher/tutor of the student

Everybody, in Web 2.0 terms: “the crowd”. For most situations, the creator will be allowed, the described person and the teacher of the student to rate statements. For testing purposes, the peer and the crowd in some selected situations to rate statements. A clear discussion about “who is allowed to read and to rate which student information” using GrappleStatements has to be undertaken in the future.

### 3.1.3 Initial Design of the Database Schema

The proposed rating system is a 5-scale rating with an RDF-triple-like datamodel (s-p-o). The choice has been made to store the user ratings in a relational database, where s denotes the ID of a statement to rate, p the predicate (IS..rated) and o (object) a digit between 1 (bad) and 5 (excellent). Additional information that needs to be stored will be the Dublin Core dimensions for creator, creation date and possibly the user’s individual rating history, which can be provided as a comma separated list that contains the digits 1-5 in order of their appearance. The following figure depicts this table as an example of a statement rating.

| s                                 | p         | o | creator     | created             |
|-----------------------------------|-----------|---|-------------|---------------------|
| STATEMENT..9B9BQkED5PD9elagclPWCj | IS..rated | 4 | USER..rmath | 2009-11-23 16:25:58 |
| STATEMENT..9B9BQkVd9HOKuLXwPv2D4j | IS..rated | 3 | USER..rmath | 2009-11-19 12:51:17 |
| STATEMENT..9B9BQk00Az3EYR2qhjutpj | IS..rated | 4 | USER..rmath | 2009-11-19 12:51:21 |
| STATEMENT..9B9BQPLMsDQjPcblykLwoj | IS..rated | 4 | USER..rmath | 2009-11-19 12:51:25 |
| STATEMENT..9B9BQPZ9J5CGcbNpsOnKuj | IS..rated | 4 | USER..rmath | 2009-11-19 12:51:31 |

Figure 2: Instantiated Database Schema for Testing Basic Ratings

From the user-statement relations of this table the average rating will be computed by adding up all the users’ ratings for a certain statement divided by the number of users as can be counted from the table. To save computing power, the result of the averaging process and the total number of ratings will be stored for every statement individually in another database table. In order to identify the related statement the statementID from Figure 2 is applied. As soon as a user submits a rating or changes a previous rating the average value and (if necessary) the number of raters will be updated. Due to this strategy expensive database requests resulting from the averaging process can be reduced to submitting ratings. Displaying certain ratings occur much more frequent and can be accomplished by a less expensive select-request on the pre-computed values.

### 3.1.4 Initial Design of the User Interface

The main user interface will be an AJAX-based widget to enter and display the individual as well as the community rating of the GrappleStatements. This widget, see Figure 2, has been developed together with the GVIZ component of WP 4.



Figure 3: User interface widget for a 5-scale rating

In the case of a fine-grained multi-dimension rating, a listing of independent dimensions as shown in Figure 3 can be used to allow for multi-dimension rating in one row.



Figure 4: Multi-dimension rating (in analogy to [4])

Of course, apart from this direct method to enter ratings via a user interface, it will be allowed to update a rating via the standardised GRAPPLE web services, to be sent via the GRAPPLE event bus GEB.

## 3.2 Design of the Reputation System

### 3.2.1 Design of the Trust Model

The GRAPPLE Reputation System is realised by a notion of “Trust in terms of reliability of the information provider”. It is expected that this notion will indirectly forward and lead to a notion of trust in terms of reliability of the author and the choice of authors.<sup>2</sup>

In the first prototype the level of trust can be set manually by the different information consumers, ( by the LMSs using information from other LMSs, GRAPPLE or GALE) by choosing different decorations and medals for the different information providers. In a second step, the trust of an information provider will be computed by counting the number of excellent rated statements in the past. This way, the reputation system will partly be based on the results of the rating system of Chapter 3. A system's reputation comprises the count of positive and negative ratings in that system's history. The concrete formula has to be set up together with all LMS partners, as soon as the first critical amount of ratings has been collected.

### 3.2.2 Design of the Database Schema

The database schema for the first prototype will be pure RDF triples that realise the relations

*/system A (as information consumer) | level of trust to | system B (as information provider) |*

while the level of trust will be a symbolic value of the range “low – medium –high”.

<sup>2</sup> It might be interesting to analyse this issue in the future since a particular author might be better at making courses that correctly assess the knowledge of a user than another author.

### 3.2.3 Design of the User Interface

The design of the user interface is the following:

- the three levels of trust that can be given to information provider by information consumer will be represented by three different medals: “gold, silver, and bronze” with red ribbons, as shown below.



- the information provider can give reputation models to the information consumers with blue ribbons, as shown below.



This user interface works for the manually given medals as well as the rating-based calculated reputation model.

|              | Moodle | CLIX | eLex | Claroline | Sakai |
|--------------|--------|------|------|-----------|-------|
| gid_usi_132  |        |      |      |           |       |
| gid_usi_173  |        |      |      |           |       |
| gid_dfki_122 |        |      |      |           |       |
| gid_tue_134  |        |      |      |           |       |

Figure 5: Reputation-Matrix: the active user adds reputation medals to information providers

It is expected that this medal-based rewarding system will motivate the different partners (LMSs, applications, users and any kind of information providers) to provide high-quality user model information to each other. If it turns out (after the first round of evaluations) that this does not work properly, alternative rewarding ideas (like monetary models) have to be analysed to ensure the exchange of user model data among different GRAPPLE partners.

An issue that finally arises is the question how to integrate both the rating and the reputation system into the rest of the GRAPPLE system. The modular approach and especially the design decisions of the basic information unit, the GrappleStatements, help a lot. The ratings can simply be included in the metadata part of the GrappleStatements. That way they are available to all GRAPPLE modules via the GEB interface. The integration of the reputation system is a bit more complex, if a certain information provider, in this case the different LMSs.

## 4 Implementation

### 4.1 Implementation Details: Rating

The implementation of the Rating System for GrappleStatements uses asynchronous Ajax requests in order to save computation time conditioned by unnecessary reloads of the whole page. When requesting a statement, the method “getRatingData()” looks up the current overall rating of that statement from the database and provides a rating bar visualising the exact value of the rating (c.f. section 3.4).

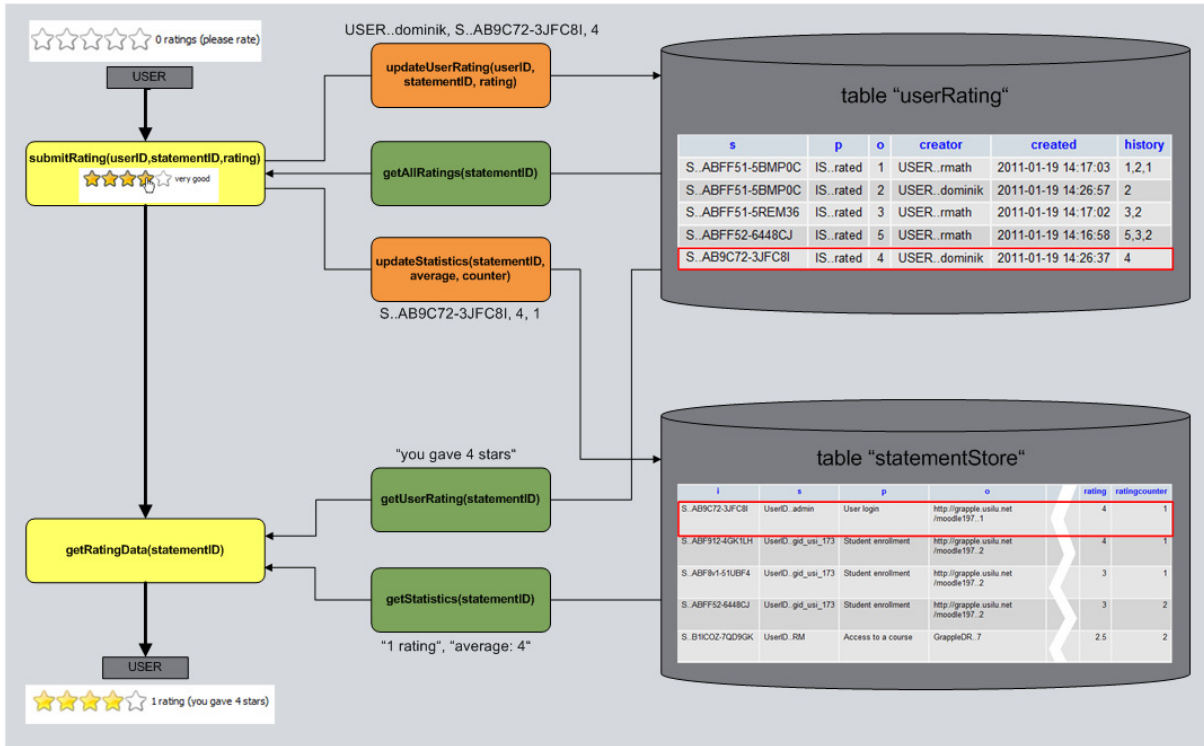


Figure 6: Data flow chart of the rating system

If the user has decided to rate the requested statement by clicking as many stars from the rating bar as he intends to rate, the method "submitRating()" will be called. According to how many stars the user selected, the corresponding rating value will be passed together with the user's ID and the statementID. The called method forwards these parameters to method "updateUserRating()", which inserts a new "IS..rated"-relation to data base table "userRating" as shown in the following example.

| s                | p         | o | creator       | created             | history |
|------------------|-----------|---|---------------|---------------------|---------|
| S..AB9C72-3JFC8I | IS..rated | 4 | USER..dominik | 2011-01-19 14:26:37 | 4       |

Figure 7: User "dominik" submitted the rating "very good" (4) for statement "S..AB9C72-3JFC8I"

Column "history" records the user's previous ratings for this particular statement and column "created" provides a timestamp of its creation. Not merely the relation which-user-has-rated-which-statement will be stored, but rather every time a rating for a statement is submitted, its total number of ratings will be counted in order to compute an average rating. To accomplish this, method "getAllRatings()" will be called, which returns a list of every user's rating for the given statement. All the ratings will be summed up and divided by the total number of ratings (arithmetic mean). The result will be passed to method "updateStatistics()", that updates the entry of the rated statement in table "statementStore" concerning average rating and rating counter, as shown in the following example.

| i                | s             | p          | o                                     | context                            | confidence | evidence                                | rating | ratingcounter |
|------------------|---------------|------------|---------------------------------------|------------------------------------|------------|---|--------|---------------|
| S..AB9C72-3JFC8I | UserID..admin | User login | http://grapple.usilu.net/moodle197..1 | http://grapple.usilu.net/moodle197 | 1          | Learninformation via GEB-ID GEB-ID..709 | 4      | 1             |

Figure 8: Statement "S..AB9C72-3JFC8I" has been rated once and has the average rating of 4.0

Storing these values in table "statementStore" creates a redundant view, as they can be computed from the statement's rating entries in table "userRating"; however, this approach saves the more expensive

computation of the average rating every time a statement is requested. Average rating and rating counter will be updated only if a new rating was submitted and can easily be looked up on a statement request.

After updating the statement's rating data in both data base tables, method "getRatingData()" will be called again to look up the new current rating of the statement which might have changed by the interaction of other users in the meanwhile. This method calls two more methods "getUserRating()" and "getStatistics()", which generate a rating bar displaying the new overall rating from tables "userRating" and "statementStore", respectively. The existing rating bar will be replaced and become inactive to prevent the user from submitting additional ratings for the given statement.

If the user returns to a statement at a later time, the rating bar will no longer be inactive. This allows him to change the rating he previously submitted; however, the individual history will be recorded in order to keep the progress as one dimension in the user model.

### 4.1.1 Rating Documentation

The following example shows a user rating and re-rating a certain GrappleStatement:

1. The statement has already been rated by another user, but not yet by the current user:



2. The current user decides to rate the statement with two stars ("ok"):



3. As soon as the Ajax-request has been processed, the new inactivated rating bar will be shown. Two users rated this statement, average rating is 2.5:



4. Now the user left the statement and returned in order to re-rate it; he selects five stars ("excellent"):



5. The user's previous rating will be discarded (but recorded in the history); only the new rating will affect the average rating:



The ratings are twofold: personal in the sense that the individual rating is stored in the user model as "act of rating", and social in the sense that the overall average of the ratings is attached to the rated statement itself that can be seen by everybody.

Figure 9 shows an example screenshot of rated GrappleStatements within the GUI environment.

| GrappleStatements Collection |                     |                    |                                       |               |                                |         |                     |    |
|------------------------------|---------------------|--------------------|---------------------------------------|---------------|--------------------------------|---------|---------------------|----|
| i                            | subject             | predicate          | object                                | x             | X                              | creator | created             |    |
| 2 ratings (you gave 2 stars) | UserID..RM          | Access to a course | GrappleDR..7                          | x0=AccessDate | x0="2010-03-01T07:42:40",x2... | Moodle  | 2011-01-12T12:24:35 | /m |
| 0 ratings (please rate)      | UserID..gid_usi_173 | Student enrollment | http://grapple.usilu.net/moodle197..2 | x0=Role       | x0="Student"                   | Moodle  | 2010-11-15T15:06:01 | /m |
| 2 ratings (you gave 4 stars) | UserID..gid_usi_173 | Student enrollment | http://grapple.usilu.net/moodle197..2 | x0=Role       | x0="Student"                   | Moodle  | 2010-11-15T15:05:01 | /m |
| 2 ratings (you gave 2 stars) | UserID..paul        | Access to a course | GrappleDR..2                          | x0=AccessDate | x0="2010-11-15T15:04:42",x2... | Moodle  | 2010-11-15T15:05:01 | /m |
| 1 rating (please rate)       | UserID..paul        | User login         | http://grapple.usilu.net/moodle197..1 | x0=IPAddress  | x0="192.168.70.136"            | Moodle  | 2010-11-15T15:05:01 | /m |
| 1 rating (please rate)       | UserID..gid_usi_173 | Student enrollment | http://grapple.usilu.net/moodle197..2 | x0=Role       | x0="Student"                   | Moodle  | 2010-11-09T09:01:02 | /m |
| 0 ratings (please rate)      | UserID..gid_usi_173 | User login         | http://grapple.usilu.net/moodle197..1 | x0=IPAddress  | x0="192.168.70.136"            | Moodle  | 2010-11-09T09:01:01 | /m |
| 1 rating (please rate)       | UserID..gid_usi_173 | Student enrollment | http://grapple.usilu.net/moodle197..2 | x0=Role       | x0="Student"                   | Moodle  | 2010-11-09T09:00:01 | /m |

Figure 9: Screenshot of rated GrappleStatements

## 4.2 Implementation Details: Reputation

The medal based manual reputation system has been implemented, where each information consumer can represent three levels of trust for each information provider by three different medals: "gold, silver, and bronze". Figure 10 shows the data flow chart of the reputation system. The active user can add and change his or her reputation medals, while at the same time inspect the reputation medals given by the other information consumers.

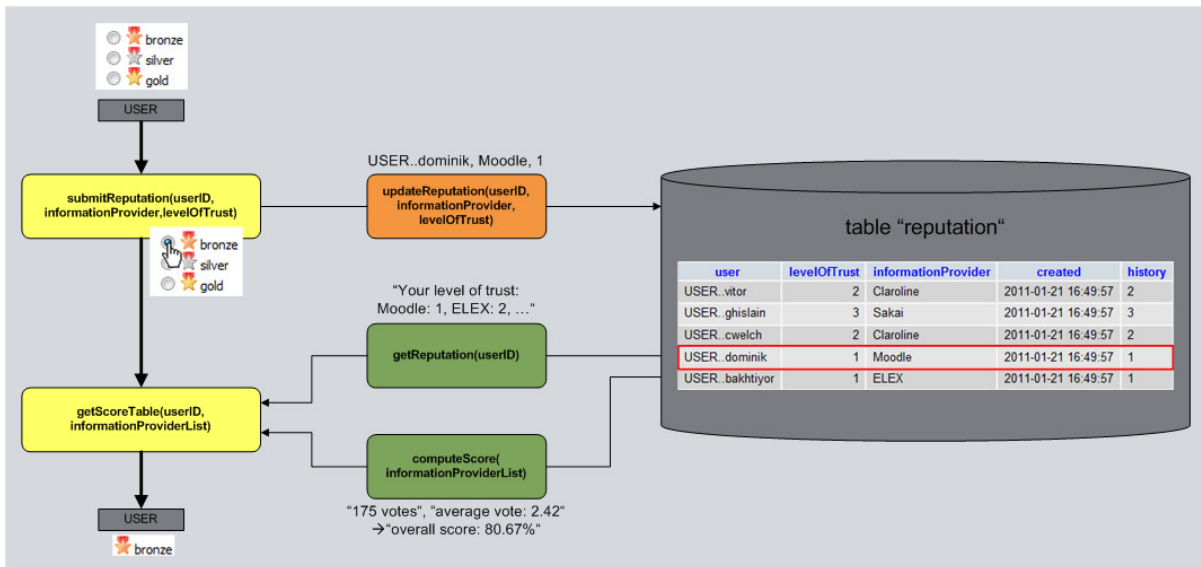


Figure 10: Data flow chart of the reputation system

The implemented database schemas are realised in analogy to the 5-star rating system in section 4.1. See Figure 11 for example entries. The history is implemented to enable a view on the progress of opinions.

| user            | levelOfTrust | informationProvider | created             | history |
|-----------------|--------------|---------------------|---------------------|---------|
| USER..vitor     | 2            | Claroline           | 2011-01-21 16:49:57 | 2       |
| USER..ghislain  | 3            | Sakai               | 2011-01-21 16:49:57 | 3       |
| USER..cwelch    | 2            | Claroline           | 2011-01-21 16:49:57 | 2       |
| USER..dominik   | 1            | Moodle              | 2011-01-21 16:49:57 | 1       |
| USER..bakhtiyor | 1            | ELEX                | 2011-01-21 16:49:57 | 1       |

Figure 11: Example of database schema entries

### 4.2.1 Reputation Documentation

A self-explaining documentation of the reputation user interface is shown in Figure 12: the active user can add and change reputation medals to information providers and at the same time inspect the reputations given by other users.

| User   | Moodle<br>100.00%   | ELEX<br>67.10%  | CLIX<br>68.71%  | Sakai<br>65.16%   | Claroline<br>33.33%   |
|--|---|---|---|---|---|
| You (rmath)<br><input type="button" value="submit"/> | <input type="radio"/> bronze<br><input type="radio"/> silver<br><input checked="" type="radio"/> gold | <input type="radio"/> bronze<br><input checked="" type="radio"/> silver<br><input type="radio"/> gold | <input type="radio"/> bronze<br><input checked="" type="radio"/> silver<br><input type="radio"/> gold | <input checked="" type="radio"/> bronze<br><input type="radio"/> silver<br><input type="radio"/> gold | <input checked="" type="radio"/> bronze<br><input type="radio"/> silver<br><input type="radio"/> gold |
| access   | gold  | bronze  | gold  | bronze  | bronze  |
| aghasaryan   | gold  | silver  | bronze  | gold  | bronze  |
| agnese   | gold  | silver  | bronze  | silver  | bronze  |
| ajaimes  | gold  | bronze  | gold  | silver  | bronze  |
| akabir   | gold  | gold  | gold  | bronze  | bronze  |
| akira  | gold  | gold  | silver  | bronze  | bronze  |
| akkuzhyna  | gold  | gold  | silver  | gold  | bronze  |
| alchua   | gold  | silver  | gold  | gold  | bronze  |
| alexandru-chitea                                     | gold  | bronze  | bronze  | bronze  | bronze  |
| alfredkobsa  | gold  | bronze  | bronze  | bronze  | bronze  |
| alrifai  | gold  | gold  | silver  | bronze  | bronze  |
| amaia  | gold  | bronze  | bronze  | silver  | bronze  |

Figure 12: Screenshot Reputation-Matrix: the active user adds reputation medals to information providers

### 4.3 Implementation Details: Integration

This section presents the entry points to the overall implementation of the GRAPPLE system and in detail also to the GUMF system.

#### 4.3.1 Integration into GRAPPLE Event Bus Technology

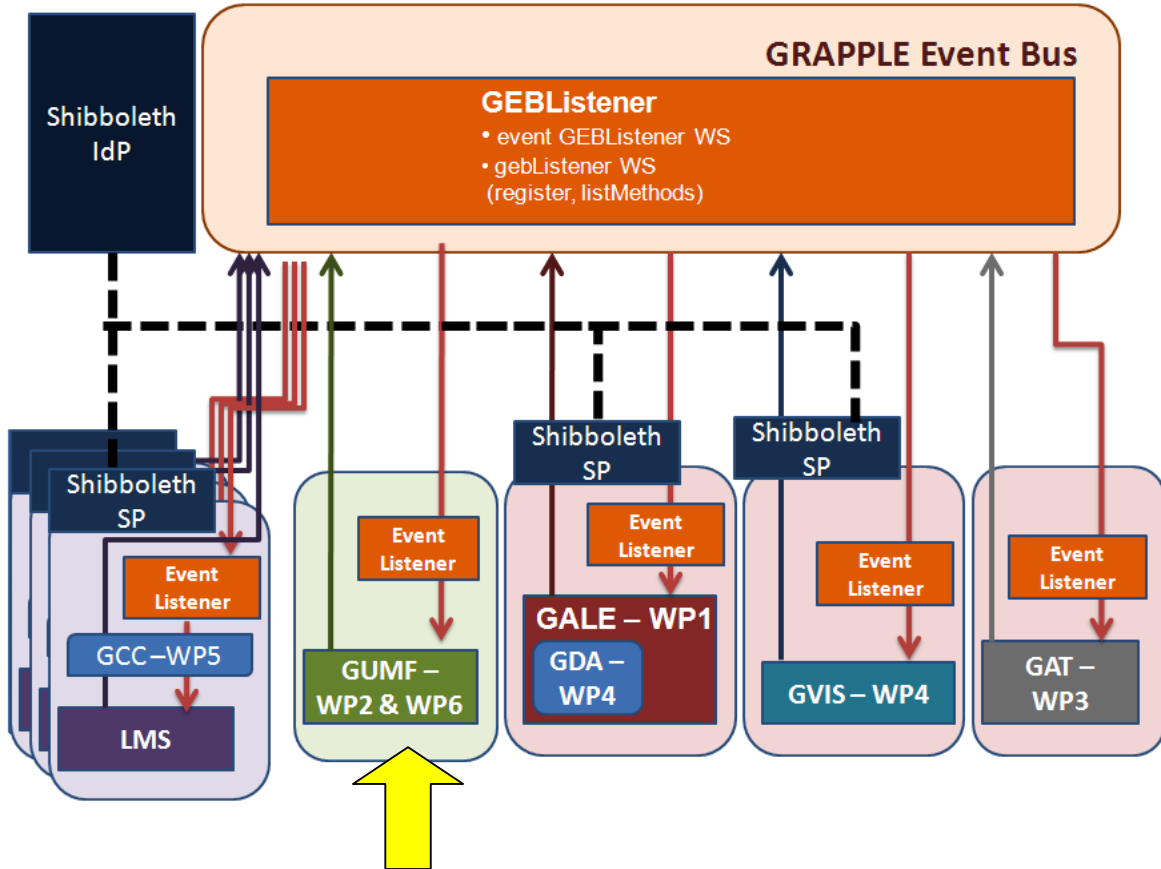


Figure 13: Event Listener via GUMF WP2 & WP6 module

The rating and reputation module is part of GUMF that has been developed in WP2 and WP6. The statements that are sent via the GRAPPLE Event Bus are enriched with the aggregated rating information. The connection is realised via a GRAPPLE event bus listener, see Figure 13.

#### 4.3.2 Integration into GRAPPLE User Model Framework

Figure 14 shows the GUMF architecture as presented in Deliverable 6.1b. Events are detected by client applications (GALE, LMS, general Web applications, event detection tools – see next subsection). Via the Client API (Grapple Event Bus, Java API or SOAP) the events are submitted to GUMF. The event storage and contextualisation components are integrated into GUMF using dataspace.

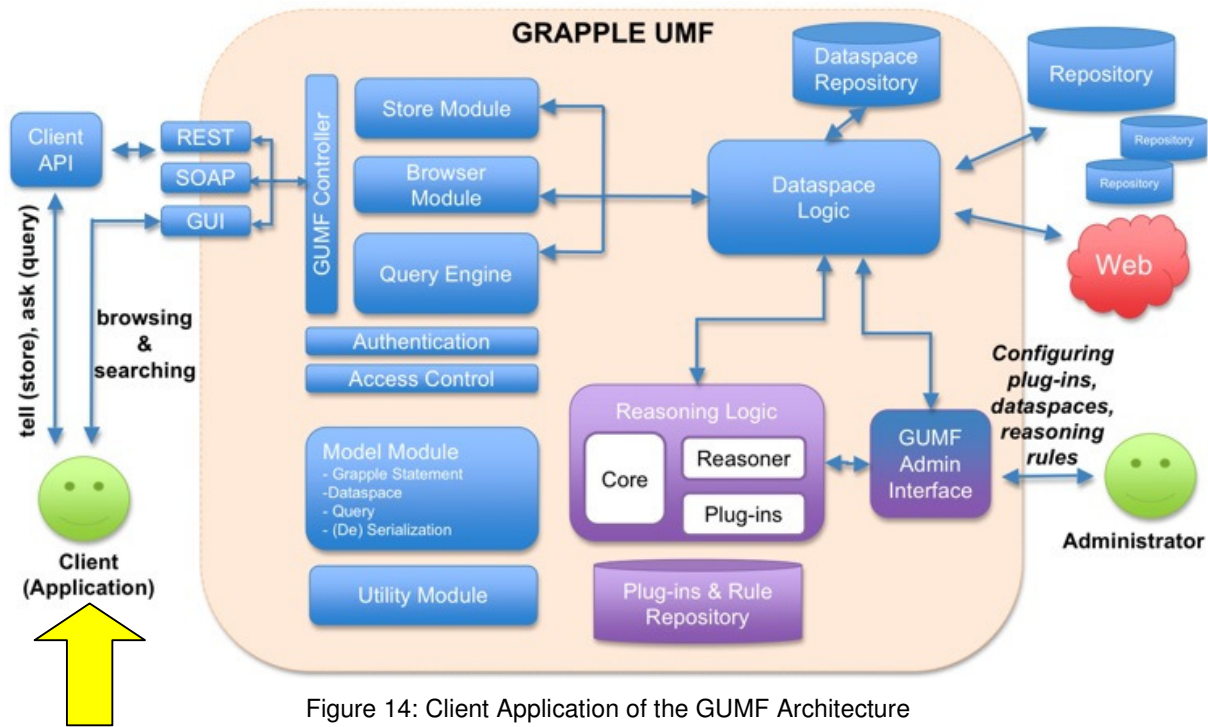


Figure 14: Client Application of the GUMF Architecture

The rating and reputation modules can be seen as direct client applications that use the GUMF controller and Client API to communicate with the rest of the infrastructure. The ratings themselves become part of the normal data within the meta-data part of the GrappleStatements.

## 5 Discussion & Summary

One major issue with structured knowledge built in a collaborative manner as given in GRAPPLE is the assurance of high-quality user-generated content, especially because there is normally no central authority giving instructions and controlling the content. The GrappleBroker can only control the syntactical appearance of GrappleStatements, but not the semantic correctness. Apparently, special ways of controlling the quality of newly created data like GRAPPLE statements are needed.

This deliverable discusses the important questions arising in this context:

- How can the generation and the control of high-quality content for the distributed TEL system be ensured?
- How can members of the learner and LMS community be motivated to actively participate, to contribute and to ensure quality?

The first step, as already realised in the GrappleStatement definition in deliverable D6.1a, is to store the identifier of the creator who is responsible for the parts of the user models. By doing so, it is possible to find out whether a user or system constantly inserts low quality content.

In the second step, the community is invited to participate in the quality assurance process of the data. For this reason, a five-star rating system for GRAPPLE has been designed, with which users can rate the statements of the user models and learner models.

In the third step it is particularly important allow for rating the raters as proposed by Noy et al. [24]. A reputation model has been introduced and designed for that.

The rewarding of users depending on the amount and the quality (determined via the ratings) of their participation seems to be a promising approach. Users identified as very active and trustworthy participants could be rewarded through GRAPPLE privileges (e.g. being promoted on the projects web site), for instance.

The five-star rating system for GrappleStatement and the medal-based reputation system for information provider has been implemented, based on literature and prominent examples in e-commerce. The overall goal is to bring this Semantic Web approach according to Tim Berners-Lee [7] to its top layer: the one of TRUST.

As future work it could be evaluated if this expected analogy between e-commerce ratings and e-learning ratings are natural.

## 6 Appendix: Selected Source Code Examples

This section presents selected source codes that are relevant to reproduce the rating functionality in a different implementation setting.

### 6.1.1 Xajax Functions

xajax is an open source PHP library for building Web-based Ajax applications. The development of asynchronous Ajax applications speeds up the user experience with the Web page at runtime.

```
<?php
    session_start();
    include_once("../xajax/xajax_core/xajax.inc.php");
    include_once("../common/inc.db-open.php");
    include_once("inc.statement_rating.php");

    $xajax = new xajax();
    $xajax->registerFunction("addrating");

    function addrating($id,$rating,$user,$table)
    {
        writeToDB($id,$rating,$user);
        updateStatistics($id,$table);
        $ratingResult = ratingContainerContent($id,$user,true,$table);

        // create a new xajax-response-object
        $objResponse = new xajaxResponse();
        $objResponse->assign("ratingContainer_$id", "innerHTML", $ratingResult);
        return $objResponse;
    }

    $xajax->processRequest();
?>
```

### 6.1.2 Statement Rating Functions

```
function displayStatusField($id,$statusText)
{
    // status field
    return "<div id='status$id' style='font: 11px tahoma, sans-serif; margin-left: 0px; width: 100%;'>".$statusText</div>";
}
}
```

```

function writeToDB($id,$rating,$user)
{
    global $objConn;

    $id = mysql_real_escape_string($id);
    $rating = mysql_real_escape_string($rating);
    $user = mysql_real_escape_string($user);

    $created = date("Y-m-d H:i:s");
    $sqlnfrage = "INSERT INTO `is-rated` (s,p,o,creator,created,history) VALUES ('$id','IS..rated','$rating','$user','$created','$rating')";
    $objResult_insert = $objConn->query($sqlnfrage);

    if(!$objResult_insert)
    {
        $sqlnfrage = "UPDATE `is-rated` SET o='$rating',history=CONCAT('$rating;',history) WHERE s='$id' AND creator='$user'";
        $objResult_update = $objConn->query($sqlnfrage);
    }
}

function updateStatistics($id,$table)
{
    global $objConn;

    $id = mysql_real_escape_string($id);

    $counter = 0;
    $sum = 0;
    $sqlnfrage = "SELECT o FROM `is-rated` WHERE s='$id'";
    $objResult = $objConn->query($sqlnfrage);
    while($row = $objResult->fetch_assoc())
    {
        $counter++;
        $sum += $row["o"];
    }

    if($counter>0)
    {
        $saverage = round($sum/$counter,3);
        if($table == "is-commented")
            $sqlnfrage = "UPDATE `is-commented` SET rating=$saverage, ratingcounter=$counter WHERE identifier='$id'";
        else
            $sqlnfrage = "UPDATE `$table` SET rating=$saverage, ratingcounter=$counter WHERE i='$id'";
        $objConn->query($sqlnfrage);
    }
}

function getStatistics($id,$table)
{

```

```

global $objConn;
$id = mysql_real_escape_string($id);
if($stable == "is-commented")
    $sqlfrage = "SELECT rating,ratingcounter FROM `is-commented` WHERE identifier=$id";
    else
    $sqlfrage = "SELECT rating,ratingcounter FROM `$stable` WHERE i=$id";
$objResult = $objConn->query($sqlfrage);
$row = $objResult->fetch_assoc();

    return array("average" => round($row["rating"],1), "count" => $row["ratingcounter"]);
}

```

```

function getUserStatus($id, $user)
{
    global $objConn;

    $id = mysql_real_escape_string($id);
    $user = mysql_real_escape_string($user);

    $sqlfrage = "SELECT o FROM `is-rated` WHERE s=$id AND creator=$user";
    $objResult = $objConn->query($sqlfrage);
    $row = $objResult->fetch_assoc();
    if($row["o"]==1)
        return "you gave 1 star";
        elseif($row["o">1)
            return "you gave ".$row["o"]." stars";
        else
            return "please rate";
}

```

```

function ratingContainerContent($id,$user,$readonly,$stable)
{
    $statistics = getStatistics($id,$stable);
    $average = $statistics["average"];
    if($statistics["count"]==1)
        $count = "1 rating";
    else
        $count = $statistics["count"]." ratings";

    $userStatus = getUserStatus($id, $user);

    $statusText = $count." ( ".$userStatus." )";

    $ratingBar = displayRatingBar($id, $user, $average, $statusText, $readonly, $stable);
    $statusField = displayStatusField($id, $statusText);

    return $ratingBar.$statusField;
}

```

### 6.1.3 Reputation Functions

```

<?php
include_once("../common/inc.db-open.php");
function writeToDB($user,$levelOfTrust,$informationProvider)
{
    global $objConn;
    $user = mysql_real_escape_string($user);
    $levelOfTrust = mysql_real_escape_string($levelOfTrust);
    $informationProvider = mysql_real_escape_string($informationProvider);
    $created = date("Y-m-d H:i:s");
    $sqlanfrage = "INSERT INTO `grapple-reputation` (user,levelOfTrust,informationProvider,created,history) ".
    "VALUES ('$user','$levelOfTrust','$informationProvider','$created','$levelOfTrust')";
    $objResult_insert = $objConn->query($sqlanfrage);

    if(!$objResult_insert)
    {
        $sqlanfrage = "SELECT levelOfTrust FROM `grapple-reputation` WHERE user='$user' AND informationProvider='$informationProvider'";
        $objResult = $objConn->query($sqlanfrage);
        $row = $objResult->fetch_assoc();
        if($row["levelOfTrust"]!=$levelOfTrust)
        {
            $sqlanfrage = "UPDATE `grapple-reputation` SET levelOfTrust=$levelOfTrust,history=CONCAT('$levelOfTrust,',history) ".
            "WHERE informationProvider='$informationProvider' AND user='$user'";
            $objResult_update = $objConn->query($sqlanfrage);
        }
    }
}

function getUserReputation($user)
{
    global $objConn;

    $reputation = array();
    $user = mysql_real_escape_string($user);

    $sqlanfrage = "SELECT levelOfTrust,informationProvider FROM `grapple-reputation` WHERE user='$user'";
    $objResult = $objConn->query($sqlanfrage);
    while($row = $objResult->fetch_assoc())
        $reputation[$row["informationProvider"]] = $row["levelOfTrust"];

    return $reputation;
}

function getScore($informationProvider)
{
    global $objConn;

    $sqlanfrage = "SELECT SUM(levelOfTrust) AS score FROM `grapple-reputation` WHERE informationProvider='$informationProvider'";

```

```

$objarResult = $objConn->query($sqlanfrage);
$row = $objarResult->fetch_assoc();
    $points = $row["score"];

    $sqlanfrage = "SELECT COUNT(*) AS counter FROM `grapple-reputation` WHERE informationProvider='$informationProvider'";
$objarResult = $objConn->query($sqlanfrage);
$row = $objarResult->fetch_assoc();
    $counter = $row["counter"];

    $averageScore = $points/$counter;
    $percentage = $averageScore*100/3;

return number_format($percentage,2);
}
function getScoreIcon($score)
{
    if($score >= 75)
        return "<img src='../img/award_star_gold_1.png'>";
    elseif($score >= 50)
        return "<img src='../img/award_star_silver_1.png'>";
    else
        return "<img src='../img/award_star_bronze_1.png'>";
}
?>

```

## References

1. Baldonado, M., Chang, C.-C.K., Gravano, L., Paepcke, A.: The Stanford Digital Library Metadata Architecture. *Int. J. Digit. Libr.* 1 (1997) 108–121
2. Bruce, K.B., Cardelli, L., Pierce, B.C.: Comparing Object Encodings. In: Abadi, M., Ito, T. (eds.): *Theoretical Aspects of Computer Software. Lecture Notes in Computer Science*, Vol. 1281. Springer-Verlag, Berlin Heidelberg New York (1997) 415–438
3. van Leeuwen, J. (ed.): *Computer Science Today. Recent Trends and Developments. Lecture Notes in Computer Science*, Vol. 1000. Springer-Verlag, Berlin Heidelberg New York (1995)
4. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
5. Allen, Christopher ; Appelcline, Shannon: Collective Choice: Rating, Systems. December 2005. – [http://www.lifewithalacrity.com/2005/12/collective\\_choi.html](http://www.lifewithalacrity.com/2005/12/collective_choi.html)
6. Allen, Christopher ; Appelcline, Shannon: Systems for Collective Choice. December 2005. – [http://www.lifewithalacrity.com/2005/12/systems\\_for\\_col.html](http://www.lifewithalacrity.com/2005/12/systems_for_col.html)
7. Berners-Lee, Tim ; Hendler, James ; Lassila, Ora: The Semantic Web. In: *Scientific American* 284 (2001), Mai, Nr. 5, pp. 34–43.
8. Appelcline, Shannon: Collective Choice: More Thoughts About Ratings. December 2006. – [http://www.skotos.net/articles/TTnT\\_/TTnT\\_198.phtml](http://www.skotos.net/articles/TTnT_/TTnT_198.phtml)
9. Wolfgang Wahlster: Kap. 1 In: Bullinger, Hans-Jörg (Hrsg.): *Trendbarometer Technik: Visionäre Produkte, Neue Werkstoffe, Fabriken der Zukunft*. München, Wien : Hanser, 2004, pp. 62–63. – [http://www.dfki.de/~wahlster/Publications/Semantisches\\_Web.pdf](http://www.dfki.de/~wahlster/Publications/Semantisches_Web.pdf)
10. Anderson, Chris: *The Long Tail - der lange Schwanz: Nischenprodukte statt Massenmarkt ; das Geschäft der Zukunft*. Hanser, March 2007. – ISBN 3446409904

11. eBay: Mehr Transparenz für noch mehr Vertrauen: eBay erweitert sein Bewertungssystem. Mai 2007. – [http://presse.ebay.de/news.exe?typ=SU&news\\_id=101149](http://presse.ebay.de/news.exe?typ=SU&news_id=101149), date: 14.08.2009
12. Fensel, Dieter ; Wahlster, Wolfgang ; Lieberman, Henry; Hendler, James: Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press, Februar 2003. – ISBN 0262062321
13. Heckmann, Dominikus: Ubiquitous User Modeling. Berlin : Akademische Verlagsgesellschaft Aka GmbH, 2006. – ISBN 3898382974
14. Recktenwald, Pascal: Ein Web 2.0 Bewertungssystem für UbisWorld, Bachelor Thesis, Saarland University. 2007
15. O'Reilly, Tim: What Is Web 2.0. September 2005. – <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>
16. Schwarzkopf, Eric; Heckmann, Dominik; Dengler, Dietmar; Kröner, Alexander: Mining the Structure of Tag Spaces for User Modeling. In: Complete On-Line Proceedings of the Workshop on Data Mining for User Modeling at the 11th International Conference on User Modeling. Corfu, Griechenland, 2007, pp. 63–75
17. Wahlster, Wolfgang ; Dengel, Andreas: Web 3.0: Convergence of Web 2.0 and the Semantic Web. In: Telekom Technology Radar II, June 2006, pp. 1–23
18. Zhang, J. and Cohen, R.: A Trust Model for Sharing Ratings of Information Providers on the Semantic Web. In: Canadian Semantic Web. Semantic Web and Beyond: Computing for Human Experience, Springer, 2006, pp. 45-61.
19. Richardson, M., Agrawal, R., and Domingos, P.: Trust Management for the Semantic Web. In the Proceeding of Second International Semantic Web Conference (ISWC 2003), Sanibel Island, FL, USA, October 20-23, 2003.
20. Brin, S., and Page, L.: The Anatomy of a Large-Scale Hypertextual Web Search Engine. Computer Networks and ISDN Systems, 30 (1-7), pp. 107-117, 1998.
21. Ding, L., Kolari, P., Ganjugunte , S., Finin, T., and Joshi, A.: Modeling and Evaluating Trust Network Inference. In the Proceeding of Seventh International Workshop on Trust in Agent Societies at AAMAS, 2004.
22. Bizer, C., and Oldakowski, R.: Using context- and content-based trust policies on the semantic web. In the Proceedings of the 13th international conference on WorldWide Web - Alternate Track Papers & Posters, WWW 2004, New York, NY, USA, May 17-20, 2004.
23. Christian Hütter, Conny Kühne, Klemens Böhm: Peer Production of Structured Knowledge – an Empirical Study of Ratings and Incentive Mechanisms. Proceedings at CIKM'08, October 26–30, 2008, Napa Valley, California, USA., pp. 827-835
24. Noy, N., Guha, R., and Musen, M. (2005). User Ratings of Ontologies: Who will Rate the Raters? Proceedings of the AAAI Spring Symposium on Knowledge Collection from Volunteer Contributors, 2005