RWTH Aachen University

LuFG Informatik 9: Learning Technologies
Chair for Communication Science

Master thesis

Quality Assurance in a Structured Collaborative Discussion System

November 18, 2015

Supervisors:
Prof. Dr. Ulrik Schroeder
Prof. Dr. Martina Ziefle
Advisors:
Dr. André Calero Valdez
Christoph Greven, M.Sc.

Felix Dietze
felix.dietze@rwth-aachen.de

Matriculation number 290734
Informatik, M.Sc.

Eidesstattliche Versicherung

_____ _____

Name, Vorname Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

_____

_____

_____

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

_____ _____

Ort, Datum Unterschrift

Contents

Abstract Popular internet discussion protocols do not scale in the number of participants. This work presents a hypergraph-based discussion system with community moderation named *Wust*. Persisted discussions of a collaborative problem solving process have an encyclopedic value for future readers. This is why discussion and post quality are of high importance. We present a definition of quality for a discussion platform on different levels. Further we implement concepts which increase quality of our platform. In the moderation system users vote on change requests by other users to approve or reject them. Users gain karma points for accepted changes. High karma allows to instantly apply changes. In this case, the community can retroactively reject them. We confirm that users find the system usable and understand discussions as hypergraphs regardless of their knowledge in graph theory, sex or age.

1. Introduction

1.1. Scalability of Today's Discussion Protocols

The communication technology has advanced exponentially in the last 40 years and now roughly connects half of the global population to the internet[1]. Today we have the Web 2.0, which is all about the creation and exchange of user generated content. Among these possibilities are social networks, media sharing platforms, blogs, news aggregators and wikis. While content creation and sharing scales quite well for many participants, discussions in direct messages or comment systems still have a scaling problem.

The simplest form of discussions are direct messages such as email. When more than two people want to discuss something, every participant needs to know what everybody else said. This means that each time someone contributes to the discussion, the message needs to be sent to all participants. The distribution can be handled by centralized servers in form of internet forums or community discussion systems. This works well for few participants. But while the distribution of information to many participants is not a problem technically, overview gets lost. The bigger the discussion becomes, the more people participate and the more complex a discussion gets; the amount of posts grows rapidly until nobody can read everything anymore.

This is definitely a problem and hinders deliberation of decisions at a large scale. Just imagine a whole nation of several million people discussing about a political decision at the same time. The technology is ready, but the protocols are not.

E-Democracy While more people create and share information online, politics only slowly benefit from this development. Most modern democracies are representative democracies, which seems to be the best approximation to a direct democracy in large countries. The concept of direct democracy worked well in history for small cities – and in some cases does today – but has never been applied to a big population. It is simply impossible to get every citizen at one place and make political decisions. In existing democracies the current state of technology only allows a trade-off between group size and depth of argument [Hil09]. Large group sizes allow direct mass voting on predefined issues, but deliberation, the discussion prior to voting, only works with a small group of people.

[1] http://www.internetworldstats.com/stats.htm

Digital communication opens new possibilities and builds a requirement for *E-Democracy*. The goal is to use the potential of mass communication to bypass the trade-off and enable large scale deliberation and decision making. Such scaling platforms could even allow to discuss global problems, involving people from all over the world. The key issue still lies in building a scalable discussion system.

1.2. Popular Discussion Platforms

Popular internet platforms address the scaling problem in different ways. They can be characterized by their discussion structure, which can be linear, threaded (tree) or a graph. We introduce some platforms briefly and take a closer look at Reddit, Hacker News and Stack Overflow afterwards.

Internet forums are usually linear. Platforms with integrated comment systems such as issue trackers or blogs are also linear, but comment systems can also be threaded. Mailing lists are threaded, because one email can respond to any other email directly forming a tree structure. In Wikipedia the articles can be discussed by editing the respective discussion page. This results in a mixture of chronological and tree-based discussions. (To our knowledge there is no popular argument mapping platform with a graph structure.)

Many platforms separate topics into isolated threads. The post in the discussion threads are either ordered chronologically or have voting mechanisms to extract the most popular content.

Next, we explain the news aggregators *Reddit* and *Hacker News*, as well as the Question and Answer (Q&A) platform *Stack Overflow* in detail and analyze their concepts.

Reddit Reddit[2] is a platform where users can post content or URLs and other users can comment and vote on it. Commenting is done in a tree with the start post being the root node. Users can up- and downvote the start post and each individual comment. A score for posts and comments is calculated as the difference between up- and downvotes. Posts are sorted by submission time and score, so that recent posts are ranked higher [Redb]. Comments are sorted by the score they will probably achieve, based on the current up- and downvotes [Mun09]. Comments responding to another comment are sorted the same way. This results in a linear visualization of a depth-first-search on the comment tree. So called subreddits are isolated areas for communities where users

[2] `https://www.reddit.com`

post content related to the topic of the subreddit. The front-page of Reddit lists the highest ranked posts of all subreddits and therefore the most popular content on the whole Reddit website. Users gain so called karma points for the votes they received on their submitted content and comments. There are also subreddits for meta-discussion about Reddit, e.g., "/r/TheoryOfReddit"[3] and "/r/ideasfortheadmins"[4].

Hacker News Hacker News[5] is similar to Reddit, but has a technology oriented community. Users can submit content and upvote the submission. Comments are also organized in a tree structure and users can vote on comments. On Hacker News both, submissions and comments, are sorted by score and time but different formulas are used compared to Reddit. Users also receive karma for votes on their submissions and comments. Downvoting is only possible on comments when a user owns a certain amount of karma [YCo].

Stack Overflow Stack Overflow[6] is a Q&A site for developers to ask and answer questions about software development. Its users can post questions and other users can post answers to that question. Questions and answers can be up- and downvoted and are sorted by different criteria which can be selected by the user. Users can also post comments to both questions and answers. Questions can be tagged and discovered by a search function. The questioner can mark one answer as the accepted one. This answer is always shown on top of all other answers. Stack Overflow is distributing reputation points for users asking good questions or giving good answers depending on their voting score. The more points a user owns, the more moderation privileges are assigned, such as editing, deleting or marking questions/answers as duplicate. There are also site moderators elected through popular vote. Stack Overflow is a part of the Stack Exchange Network[7], which consists of many Q&A sites on different topics which work the same. Stack Overflow itself can be discussed on the "Meta" site[8].

1.3. Concepts and Problems

The presented discussion platforms work well and scale with thousands of users. Considering the details of their concepts, many interesting positive and negative aspects can

[3] https://www.reddit.com/r/TheoryOfReddit
[4] https://www.reddit.com/r/ideasfortheadmins
[5] https://news.ycombinator.com
[6] https://stackoverflow.com/
[7] https://stackexchange.com/sites
[8] https://meta.stackexchange.com

be identified. We first investigate the structure and then focus on quality, redundancy and overview.

Question and Answer Systems Q&A Systems solve some scaling issues by sacrificing free form discussion. In fact, a lot of internet communication fits into the one-Question-many-Answers model. These are questions where an answer already exists. But they fail in modeling the development of an answer or solution to a problem. As a result, many questions are closed by moderators with the reason of not fitting the question-answer-style. There is also no room for a discussion starting with criticism or an idea. In contrast, on the "Meta" site, users are posting issues and feature-requests as questions and tag them appropriately.

To our knowledge Q&A-systems were the first popular systems introducing the concept of classifying posts but treating them differently. There are questions, answers, and sometimes comments on them. The popularity of Stack Overflow shows that users can handle the concept of structure and treat specific classes of posts differently. There are many academic argument-mapping systems that try to classify even more posts and give a complete structure to a debate. More on this topic can be found in the work of Karoff [Kar15].

Threaded Discussions Threaded systems model discussions as trees of comments. One comment can have multiple responding comments. Users can vote on comments to create an order for the children of a comment. This helps to display the tree with a depth-first-search algorithm. The result is a view with a critical comment-path first, followed by the rest of the depth-first traversal. This view has some drawbacks. First, the subtree following the second-voted comment can be far away if the first comment gets a lot of answers. Second, the critical path is convenient to read, because it is a chain of upvoted comments answering each-other. But it can go into arbitrary depth, down to a level which might not be interesting to the reader anymore. By being displayed at the top, the first comment-subtree gets more comments, while the other first-level comments are left behind. This is a so called *rich-get-richer*-problem. Another problem is that quality is confused with popularity, because most users only upvote. This effect happens on all levels of the tree. At this point the chronological order of posts does not matter. The only entry point is the highest voted comment on the first level with the critical path. This gives the illusion of being the most important part of the discussion.

Back Links and Quotes In linear systems, old or big discussions are difficult to work with. Responding only appends to the end of the list of posts. To refer to another post in the discussion, one has to copy a part of the post one is referring to and mark it as a quote. There is usually no back-link from the quoted post to the new post. Hence, reading the discussion from the beginning does not give a clue, that an early posted question is answered in the end. The same applies for mailing lists. Some systems, such as the issue tracker on GitHub, provide a back-link-detection to avoid this.

In Reddit and Hacker News, responses are directly connected to the respective comment. This avoids the quoting and missing back link problem. Still, in the depth-first-view not all children are visible nor are there any indicators for the number of children. Additionally, when there are many comments it is difficult to find corresponding comment of a response.

Duplicates Duplicate posts are a problem. When posting something that has been posted before on Reddit or Hacker News, it is called a *repost*. Users are encouraged to do a search before creating a post to avoid reposts. But with a high volume of incoming posts, reposts are the norm [Gil13]. They are either downvoted, ignored or deleted, regardless of their attached comments. These platforms do not support the merging of posts. A repost in another subreddit is called a *cross-post* (or *x-post*). These cross-posts do not share comments and therefore create the possibility for duplicate comments. Some redditors consider cross-posting as karmawhoring if a post is not explicitly marked as cross-post. Stack Overflow can merge questions, but encourages removing duplicate answers. In this case, the comments on removed answers are lost, too. Also, in Q&A-systems, one answer cannot solve two similar questions, so that the answer must be duplicated and posted on both questions.

Moderation In internet forums usually the site administrator deletes or edits posts. In bigger forums the administrator distributes these permissions to users moderating a subforum. Such a system relies on the moderators reading everything. This can be a very difficult and time consuming task if the forum has many users. Usually there is also a button to report a post to the moderators.

On Reddit, the creators of subreddits become moderators [Reda] and can designate other redditors to become moderators as well. Criticism indicates that the owner of a subreddit is not necessarily a good moderator for the subreddit [III14].

Stack Overflow uses a different approach. Every user is assigned a numerical value called reputation. When posting high quality answers or questions and other users vote on

them the user's reputation increases and the user gets more permissions on the site. Moderation actions include editing of questions and answers and closing answers.

Voting and Karma The karma points on Reddit are a gamification mechanism with the original intention to motivate users to create high quality content on the platform. But it turns out that users are even motivated to gain karma by all available means. This results in users choosing content which most likely appeals to the lowest common denominator among all other users. This in turn encourages repetitive, instead of innovative or controversial content [Ric14].

On voting platforms it is often not clearly defined, what up- and downvoting actually means. Some associate it with quality, others with agreement. There were extreme cases on Reddit where users posted simple yes-no questions and received an immense amount of upvotes. The upvotes were meant as agreement and not the quality of the question. Because of this effect Reddit does not distribute karma for own content (self-posts) anymore [Redc].

Comment voting hinders the creation of discussions with overview. When someone has something interesting to say on a popular post on Reddit, he has to post it as a reply of the top comment, otherwise nobody will ever notice it [Ric14].

We conclude that voting is at least a mixture of opinion, quality and popularity. Even more, users rarely vote regularly, as can be found in our study in chapter 5. They prefer upvotes over downvotes and only vote on special posts.

Overview and Redundancy We expect that for large scale discussions the number of different opinions and perspectives for one topic is limited and much smaller than the number of participants. When all participants can write at the same time, they naturally create redundancy.

Also in chronologically ordered systems there are two entry points to a discussion: The newest and the oldest post. Everything in-between has to be discovered with a linear search starting from one of these points.

When responding to an old discussion in linear systems, the thread is shown on top of the list again in order to signal that there was activity in the thread. In Reddit and Hacker News, new comments in an old discussion present no hint of activity. When looking at past time frames, only the highest voted discussions can be found. The others appear to be lost. In most cases it is better to start the same discussion again instead of working on the old one. This again produces redundancy at the post and

discussion level. Sometimes creators of a thread update the start post to reflect the current state of discussion.

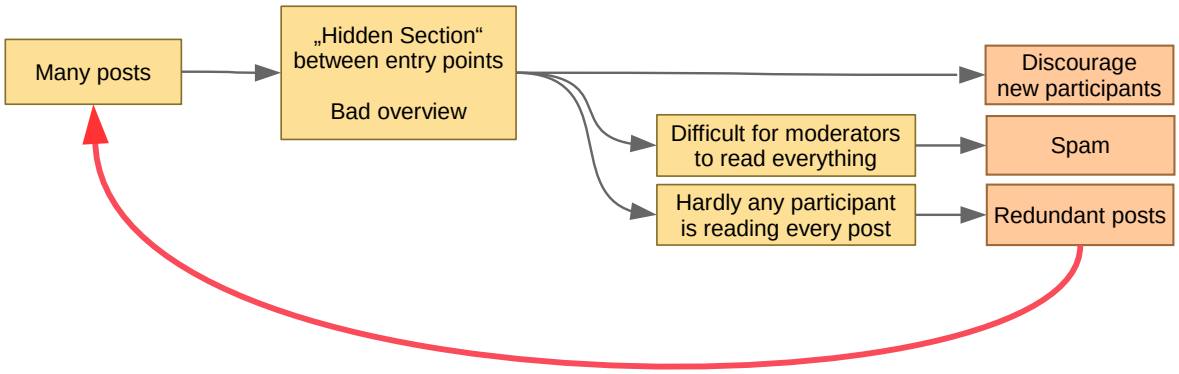


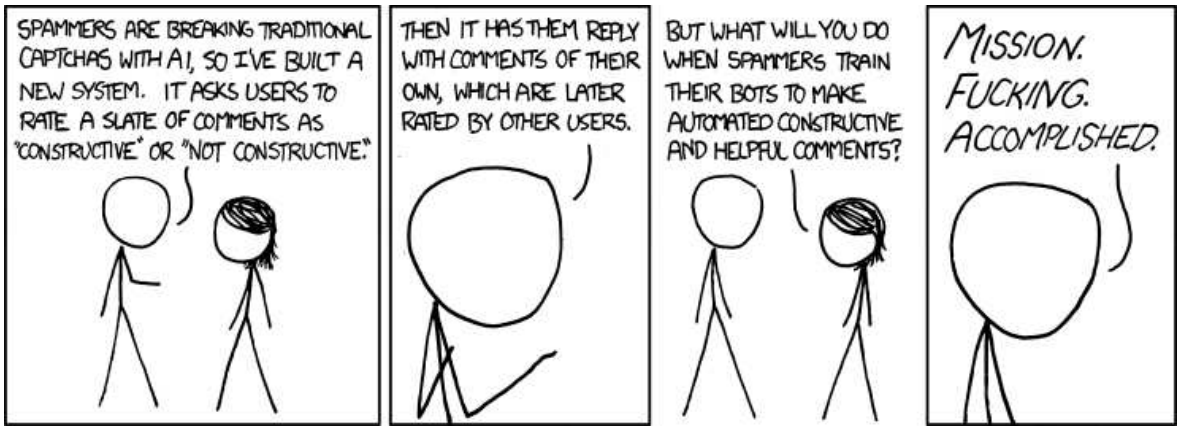
Figure 1: Discussions do not scale, because many participants, high topic complexity and big discussion scopes result in more posts being created. When these posts are not easily discoverable and the presentation has a bad overview, it leads to redundancy which produces a feedback loop and is creating even more posts.

This creates a feedback loop, as can be seen in Figure 1. Many participants, a big discussion scope and high topic complexity naturally result in many posts being created. This is bad for the discussion overview and leads to duplicate and redundant posts. Due to the huge amount of posts and lack of different entry points, participants cannot read every post anymore. This is especially bad for new participants joining a discussion. As a result, more redundant posts are created, contributing to the feedback loop. At some point the discussion is too big and cluttered with redundancy, which indicates a reduced discussion quality.

1.4. Overview of our Work

As described earlier, scaling is mostly hindered by several quality problems on different abstraction levels. In this work we address some of the aspects contributing to the feedback loop. In collaboration with Johannes Karoff [Kar15], a hypergraph-based discussion system called Wust was developed. It tries to highlight redundancy to prevent creating it in the first place. We present a definition of quality on different levels and explain how our approach contributes to improving quality. The description of the approach and concepts of our system can be found in chapter 3. Beginning with a description of the discussion system, we continue with a definition of quality and describe how different concepts of the system should improve quality. Wust is a discussion website with community moderation, voting and graph visualization. The implementation is described in chapter 4. It also covers the discussion model and a graph database

abstraction to map the high level model to the database. Our user study can be found in chapter 5. It starts with a description of our hypotheses and evaluates them with two user test and a questionnaire. In the end we propose an outlook for future work and give a conclusion in chapter 6.



"And what about all the people who won't be able to join the community because they're terrible at making helpful and constructive co– ... oh." (`http://xkcd.com/810`)

2. Related work

Since more and more people are connected to the internet, the need for scaling discussion solutions arises. There are different problems in current online platforms preventing scaling discussions in the number of participants. There has been a lot of research. We present works from the field of large scale communication, quality and popularity analysis on existing online platforms including voting and ranking, moderation and reputation systems and collaborative tagging.

2.1. Large Scale Communication

With our current communication systems, large discussions do not scale. This is a problem which prevents us from having real political deliberation and forces us to build a hierarchy of people to make nationwide and global decisions.

Last et al. [Las14] describe the concepts of a global brain, an analogy of distributed and decentralized global information flow similar to the human brain. Global problems are complex and can only be solved by collective human intelligence. This could be possible with the help of digital communication systems.

Klein et al. [Kle11] describe the deliberation scaling problem in depth: Social media provides bad opportunities for interacting on a massive scale. It typically generates poorly-organized, unsystematic and highly redundant contributions of widely varying quality. Large-scale argumentation systems represent a promising approach for addressing these challenges, providing a simple systematic structure that radically reduces redundancy and encourages clarity. Their proposed solution is a large scale argument mapping website which lets users construct argumentation trees of *issues*, *ideas*, *arguments* and *comments*. They tested it with 220 students discussing the use of bio-fuels in Italy. After three weeks they had nearly 3000 structured posts (ideas, arguments) and 2000 comments. It turned out that two dedicated moderators were enough, because the students were able to structure their own posts correctly into the argument map. The 2000 comments suggest that the discussion model could be improved, since a comment can be anything.

The concept of E-Democracy needs scaling discussion systems for political deliberation. Hilbert et al. [Hil09] describe that in current democracies there is always a trade-off between depth of argument and group size (figure 2). Representative democracies scale well but reduce the contribution of the citizen to a yes-no vote on a political direction. On the other hand direct democracies let the citizens deliberate over political

discussions but only work in small group sizes. The internet allows fast votings on political decisions for a larger population but still suffers from the trade-off.

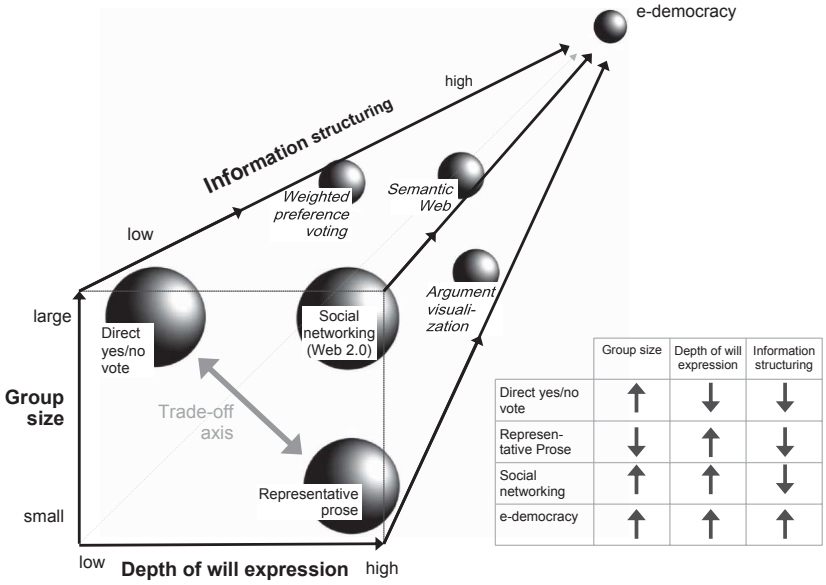


Figure 2: The challenge in E-Democracy: Current democracies have to make a trade-off between group-size and depth of will expression. Structuring discussion systems could solve this issue and allow both at the same time [Hil09].

2.2. Quality, Voting and Ranking

In current online discussion platforms the huge amount of content is handled by ranking and sorting to be accessible. The sorting criteria vary and relate to the field of content quality. We first look at a quality definition and then focus on approaches for improving quality, ranking content and its effects.

Many works on online platforms refer to the terms *post quality* or *site quality* and define this quality by the voting behavior of users. On the other hand, the term *fitness for use* describing *information quality* is widely adopted in the quality literature. Wang et al. [WS96] propose a framework for data-quality to identify dimensions of the concept, which can be seen in figure 3.

Diakopoulos et al. [DN11] analyzed commenting systems in news organizations. They investigate in the impact of low quality comments on both users and journalists, explain how individual differences in reading motivations can impact perceptions of quality. They also describe flagging, moderation, and engagement as strategies for improving quality.

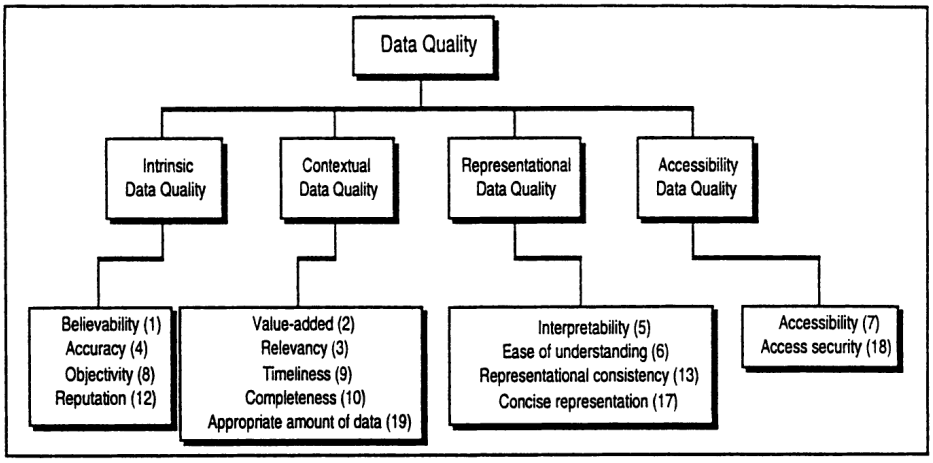


Figure 3: The Conceptual Framework of Data Quality by Wong et al. [WS96]

Previous works show that bug reports need quality improvement, with more complete and concrete descriptions of failures, and less uncertainty [Sun11] [Bet+08] [SRC08]. Lotufo et al. investigate the use of game mechanisms like in the Stack Overflow moderation system to address quality problems in bug trackers [LPC12].

Paskuda et al. [PL15] show that anonymous answers on Quora are not worse than posts by other users and did not receive significantly fewer upvotes. Answers are not shorter but are even slightly longer. Also in contrast to expectations, they were not less polite.

There are lots of works on the prediction of high quality questions and answers. Anderson et al. [And+12] are doing this using the unit of analysis being a question and its entire set of answers. Ponzanelli et al. [Pon+14] try to classify good and bad questions to automate the review process. Vasilescu et al. [Vas+14] show that the R-Community shifts from mailing lists to the Stack Exchange network and that Stack Exchange provides faster answers.

The results of Ahn et al. [ARH07] show that playfulness is an important aspect of enhancing user attitude and behavioral intention to use a site. They also found that web quality, categorized into system, information, and service quality, had a significant impact on the perceived ease of use, playfulness, and usefulness, and consequently, that it encouraged website use in the context of online retailing.

When content can be discovered by categorization, there is usually more content to display than can be shown to the user. This is even more extreme, when there is no categorization system. Therefore, the system needs to start somewhere and provide an initial selection to the user and guide him through the rest. A solution is to define an order on the content. Many systems have the goal to define this order in terms of

quality and/or popularity of content. But the problems lie in the definition of quality and popularity and how to calculate or at least approximate these values. Such calculations can combine different metrics which can be contained in the content itself, metadata like submitted time or gathered through external resources, like votes or interaction/relation of users. Depending on the selected combination of metrics different problems can be solved and created.

When sorting by up and downvotes the score function which defines the order is important. Zhang et al. [Zha+11] describe the problems of trivial approaches like difference and proportion. They present a set of axioms which reflect their intuition about what a reasonable score function should be like and provide alternatives which satisfy those axioms. Proportion based approaches correlate the most with perceived rating of up and downvotes [CDL14].

When sorting content by popularity and presenting the users a list with the most popular content on top a *rich-get-richer* phenomenon arises. The most popular contents are more visible on average, than their less popular counterparts. This phenomenon has been analyzed in several works.

Mills [Mil13] found in an experiment on Reddit that the difference in quality for a post on the front page and in a subreddit is much smaller than its score suggests. Reddit's voting system produces a highly skewed distribution of scores. He also found that 'good' posts can receive low scores and 'bad' posts can appear in high-visibility locations. Also, scores presented next to the scored items influenced participants' perception of their quality. This suggests that users have learned a relationship between score and quality.

Displaying the popularity or quality makes a difference in voting behavior [WH08]. Furthermore, the findings of Muchnik et al. [MAT13] suggest that social influence substantially biases rating dynamics in systems designed to harness collective intelligence.

Stoddard et al. [Sto15] investigate if a rich-get-richer phenomenon implies that there is a distorted relationship between quality and popularity on these platforms. They define quality as the total score an article would have received if articles were ranked randomly and no social signals were displayed about the articles. They find that popularity on Reddit and Hacker News is a relatively strong reflection of intrinsic quality.

Hacker News and Reddit have the problem that articles on their front page get a much higher exposure than every other post. Only few users look at the second page or check out the *new* section of articles. Luu [Luu] proposes the idea of adding a small amount of

random noise to the rank of an article to smooth out the discontinuity between making it onto the front page and fading away in obscurity.

Gilbert showed that voting sites like Reddit suffer from underprovision: too many people rely on others to contribute without doing so themselves. Notably, users of Reddit overlooked 52% of the most popular links the first time they were submitted. This suggests that many potentially popular links get ignored, contradicting the site's core purpose. Many links that could reach popularity never get noticed, beyond the duplicate submissions examined in their analysis [Gil13].

Cheng et al. [CDL14] analyze the feedback effects of users voting on each others content on the user level. They find that negative feedback does not increase the contribution quality of users, it makes it worse. In contrast, positive feedback does not let users create higher quality content. On the other hand a Study by Cosley et al. [Cos+05] finds that oversight – a review of user contributed content by other users – increased both the quantity and quality of contributions while reducing antisocial behavior. At the same time, peers were as effective at oversight as experts. In another study they found that showing people their unique qualities relative to a topic increase contributions [Lud+04]. Additionally they find that community-maintained artifacts of lasting value will reach the same final value whether contributions are reviewed before or after they are made available to the community [Cos+06]. Systems that require certain actions from its members can use intelligent task routing to increase contributions. Even simple algorithms have large effects. Algorithms based only on the community's needs are less likely to interest members than algorithms that consider a person's knowledge and ability.

2.3. Moderation and Reputation Systems

Since users can post freely on most web 2.0 sites, some can create content which is unwanted by a group of other users. Examples and solution attempts are spam, vandalism, illegal content, wrong information, explicit content, mobbing, trolling and many more. There are different technologies to fight this. Some rely on algorithms to filter unwanted content. Others have human moderators who have the permission to delete and modify content sometimes with the help of algorithmic approaches.

When looking at moderation systems, it is interesting to analyze their performance. There are in general two kinds of systems: designated/elected moderators and community moderation. The problems of a fixed group of moderators is that they can miss

content. If a lot of content posted, the number of moderators needs to be adjusted properly. If they are designated, the community cannot do anything about misbehavior. The moderator's actions need to be watched and corrected somehow. Slashdot has the approach of having two kinds of moderators. Comment moderators and meta-moderators who moderate the comment moderators [LR04].

Another approach is community moderation. Every user at every time can become a moderator. By doing positive actions on the site they can earn reputation which in turn gives them more permission to have advanced features like moderation.

On Stack Overflow the moderators are criticized by newbies whose questions are closed and they cannot do anything[9]. The older Stack Overflow users complain that the newbies are not taking time to read the FAQ and do their research[10].

Lampe et al. [LR04] investigate if a system of distributed moderation can quickly and consistently separate high form low quality comments in an online conversation. Their analysis of the site Slashdot.org suggests that this is true, but that important challenges remain for designers of such systems.

When users interact on the internet in most cases they have never met and interacted before in person. Reputation systems propose an indirect way of providing trust information between users. This can be done in many different ways. Hendrikx et al. [HBC15] provide a survey of reputation systems and develop a taxonomy for categorization. In the implementation section 4 we categorize our system with this taxonomy.

De Alfaro et al. [AA14] built a system called *WikiTrust* which estimates the quality for text passages in Wikipedia. They calculate a reputation score for each author which is high when the created content of the author is not removed by other authors. Text passages which have not been modified for a long time or have been created by high reputation authors are defined as high quality.

Chatterjee et al. [CAP08] extend the WikiTrust reputation algorithm with features for preventing several kinds of attacks to gain reputation without contributing valuable content.

[9] https://sergworks.wordpress.com/2012/09/26/why-stackoverflow-sucks
[10] https://meta.stackexchange.com/questions/9953/could-we-please-be-a-bit-nicer-to-new-users

2.4. Taxonomies, Folksonomies and Collaborative Tagging

To deal with the immense amount of posts in popular online platforms, ranking and sorting only provides one entry point to discover the content. Categorizing posts with tags is a widely adopted approach to make big amounts of content organized and searchable.

When looking for information in a library, different areas or categories of books are maintained by the library. This categorization vocabulary is called a taxonomy. One can rely on the fact, that the library has a controlled vocabulary to categorize books. A controlled vocabulary has several advantages [MM06]: It avoids information discovery problems with synonyms, homonyms and hypernyms. This works in small-scale with a limited amount of books, where the amount of new incoming books is manageable by a dedicated group of people or by the publisher. But this approach does not scale well with the amount of content.

On open collaboration platforms the problem is visible. Any content can appear, so it is natural that there is always content which does not fit in a predefined vocabulary. It needs to be extended to cover the classification of new content. Someone has to maintain the vocabulary. If there is a dedicated group of people, a submitter has to wait for a change to be apparent. Also, the group may not have the knowledge to create the vocabulary in specific cases.

One approach to overcome the problem of vocabulary maintenance is to let the users decide on the vocabulary, a so called folksonomy, where arbitrary strings can be used to categorize content. It brings all disadvantages which do not arise in a controlled vocabulary. Systems need to deal with these if they want their users to find the existent content efficiently, or even find it in the first place.

Sen et al. [Sen+07] state that because users often spread tags they have seen, selecting good tags not only improves an individual's view of tags, it also encourages them to create better tags in the future. They further suggest that tagging systems that support positive ratings should also support negative ratings.

Suchanek et al. [SVG08] find that the more popular tags of a page tend to be the more meaningful ones. They conclude that roughly one third of the tag applications may be induced by tag-suggestions. They also suggest that making tags clickable will not bias the tag applications, but benefits those users who prefer clicking over typing.

Heymann et al. [HG06] propose an algorithm to extract a hierarchical taxonomy of tags from a huge set of tags annotating objects.

We build our concepts of Wust on the findings in quality ranking and take ideas and criticism from community moderation systems. We also implement mechanisms to address the problems with collaborative tagging systems. The next section describes these concepts in detail.

3. Quality Assurance in Wust - A Hypergraph-based Discussion System

In this chapter we describe the concepts of the hypergraph-based discussion system Wust. We give an introduction to the discussion system, define quality on different levels and analyze concepts and their quality contribution in detail.

3.1. Introduction to the Concepts of Wust

Wust is a website, where users can write posts. A post consists of a title and an optional description. Posts can be connected to each other and form a directed graph. A directed relation $A \rightarrow B$ means that A responds to B. Posts can also be connected to other connections. These connections can be seen as hyperedges, relating two or more posts to each other.

There are two views on the graph. One is an interactive visualization of the whole discussion graph with nodes and edges. In this view users can respond, edit and connect posts to other posts or connections. The second view focuses on one post and its metadata and shows the incoming connections on the right and outgoing connections on the left. Users can read the whole description, respond and edit in this view as well.

There are two kinds of tags: Context-tags and classification-tags. Context-tags categorize posts to a topic or project. These tagged posts are entry points to discussions. Contexts can inherit from multiple other contexts. This means that a post tagged with a child-context can be discovered in the parent-context. Users can create new contexts and edit their inheritance. Each context is displayed with a pseudo-random color. The second kind of tags are classification-tags, which describe the relationship either between two posts or a post in relation to a context. Classifications are a fixed set and cannot be created by users. Classifications have fixed colors and symbols. Every post is displayed with the color and symbol of its outgoing classification relations. See figure 7 for the fixed set classifications and some examples of context-tags.

Users collect karma per context to receive more moderation permissions in that context. They can edit the posts of other users. Every post has an edit-threshold. If the editor has not enough karma the change is submitted as a request. So other users can vote on it to approve or reject it. Otherwise, it is applied immediately and other users can vote on it afterwards to revert or validate it. Editors gain karma for approved and validated

changes and lose karma for rejected and reverted ones. Figure 5 describes the process with a flowchart.

There is a view called *Moderation*, which shows one open change request or past change to vote on. Every user can reject, skip or accept changes of other users. After rating one change, another one is displayed. In this view users can contribute to the system by just clicking buttons.

Users can vote on posts with respect to a context or another post. In this way they can express that a post has a high quality regarding the relation. The author then gains karma in the specific context. When voting with respect to another post, the author gains karma for all the contexts which can be found upwards in the post hierarchy. The voting on a context has effects in two ways: It describes the posts quality to rank the post in the context and it tells how well the context-tag fits to the post in relation to other contexts-tags on this post.

3.2. Definition of Quality

In order to see how Wust can address some of the problems mentioned earlier, we need to define what quality means in our case. We define quality measures at three different levels:

- **Post Quality:** The quality of a post viewed in isolation. This needs to be separated into **Information Quality** and **Presentation Quality**. When a post is worth reading, it is of high information quality. It is worth to be archived and read in the future, because it contains knowledge. Wrong information wastes readers time and is of low information quality. Also, spam and vandalism do not contribute to the discussion and are considered to be of low quality. When it is obvious what a post is about at first glance, it is of higher presentation quality compared to the case when it is not obvious. If a post is redundant or obsolete, it is of low presentation quality. Also, typos, grammar and punctuation mistakes make a post difficult to read and are signals for low presentation quality. A post which is not easily discoverable when searching for it or a wrongly categorized post are both of low presentation quality.
- **Discourse Quality:** The quality of the discussion as a whole. This is about the content and the structural and graphical representation. High quality discussions are a summary themselves and reflect the current state of the discussion. There are no duplicate or redundant posts and no confusing relations. Participants

have a good overview of the discussion. It is easy for new participants to join the discussion. Parts of the discussion are referring to relevant parts of the current or other discussions. Posts containing several statements, which could be expressed with separate posts in the current discourse model are also considered to be of low quality.

- **Platform Quality:** The quality of how the platform makes information accessible. Users can find exactly what they are looking for in an efficient manner. When temporal information becomes obsolete, it should be hidden. Timeless information instead should be archived and well discoverable. The system should be easily usable for non-experts while providing a powerful interface for experts. When the same discussions exist multiple times, this is a sign of low quality. When the quality of posts and discourses is mixed with other quantities such as popularity (rich-get-richer) or opinions, the platform quality is negatively affected.

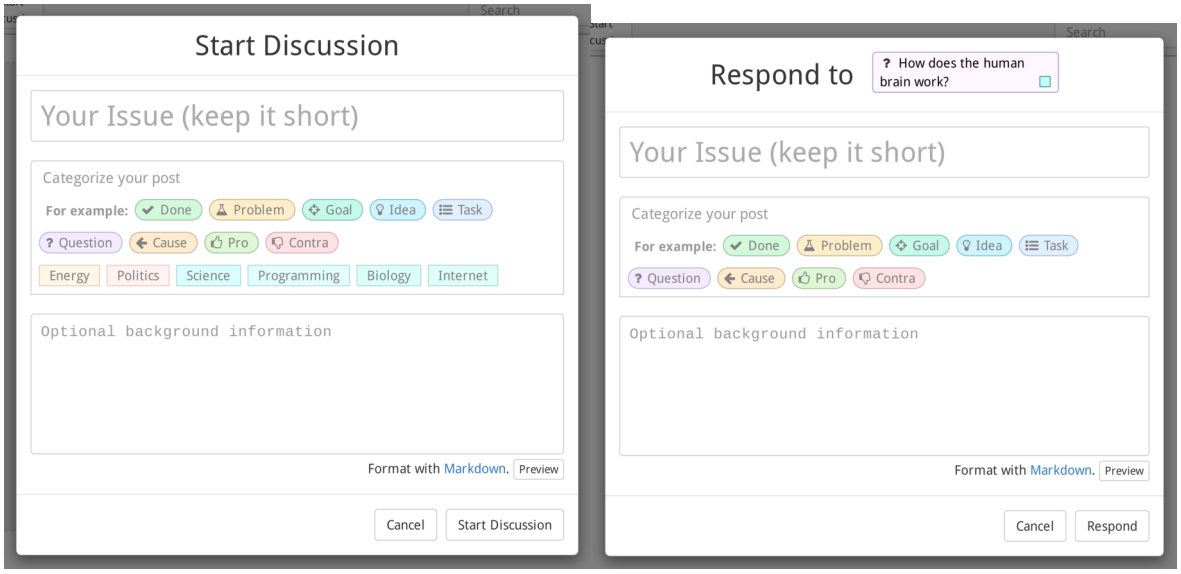
In the next sections we will show how the concepts of Wust address quality at different levels.

3.3. Title with Optional Description

In traditional discussion systems users need to fill 1 or 2 form fields. The content itself and sometimes a title which can be optional or mandatory. In the mandatory case it is most of the time prefilled with the title of the thread (internet forums) or subject (email). In email software it only helps to group many emails with the same subject and display them as one thread. Since in a graph- based system the posts are connected with each other already, this is not needed. So the title in most cases is obsolete. Answers in Stack Overflow and comments in Hacker News and Reddit completely eliminated it. Only threads have a title which is chosen by the thread opener to be able to display a list of start posts. So in the best case the title is the essence of what the posts content is about.

To get an overview of a complete discussion we need to display many posts in a small space. To solve this we decided that in Wust posts need to have a mandatory title of maximum 140 characters and an optional description. This is different than in traditional systems. The respond/start-discussion form suggests you to write the title first, and, if you have more to say you can give an optional background information. With

this requirement we always have a short summary of what users said and can display only this information in graph and list representations. The form to start a new discussion can be seen in figure 4.



(a) Start Discussion Dialog (b) Respond Dialog

Figure 4: The dialogs to start a new discussion (a) and respond to a post (b). The title input field is single line but becomes multi line when needed. It is limited to 140 characters and mandatory. The background information is optional. This design encourages writing the summary of the issue first and avoids salutations. It also puts the tag suggestions in a prominent position and encourages to at least classify posts.

Since the title field is very prominent and not multi-line, it should encourage users to avoid salutation formulas and thank-you phrases. In most cases these are not necessary to solve a problem or answer a question. On the Stack Overflow platform, many users still write salutations, but the moderators are encouraged to remove them[11].

This supports post quality, because it is clear what the post is about when looking at the title/summary. It also supports discourse quality, because it allows to display the posts graphically in an overview, without the need to display it completely.

3.4. Collaborative Tagging

When having a *folksonomy* (collaborative tagging system), users can tag content with arbitrary strings. There are several problems that can arise. When users cannot find

[11] https://meta.stackexchange.com/questions/2950/should-hi-thanks-taglines-and-salutations-be-removed-from-posts

the post they intend to create, they will possibly create a redundant post. Therefore, it is important that common problems with folksonomies are addressed by the system.

Multiple Inheritance When one tag is the hypernym of another tag, sometimes users assign both tags to one post. Example: There is a tag *Animals* and a tag *Dogs*. *Animals* is a hypernym of *Dogs* and *Dogs* is a hyponym of *Animals*. When a post is about Animals, it should have the tag Animals. When a tag is about dogs, it should have the tag Dogs. But dogs are animals and it should be possible to find the post when searching for posts with animals. Wust implements multiple tag inheritance. This means that there is an inheritance relation between *Dogs* and *Animals*. So if a post is tagged with *Dogs* it will be listed in a Search for Animals. Since a tag can have multiple hypernyms, e.g. posts talking about *Object Relational Mappers* are found when looking for *Databases* as well as *Programming*. This removes clutter in the tag-section of posts, especially when there are deep tag hierarchies.

Context-Tags mark entry points to discussions. A discussion can be seen as the induced subgraph of every post reached by traversing all relations in reverse direction from an entry point. Therefore, discussions can contain other discussions. This is natural, because sometimes in a discussion an excursion is branching off and forming a completely new discussion.

Synonyms Users give the same name to different tags with the same meaning, like for example *destiny* and *fate*. It also happens that some users use plural and others use the singular form of a word (*Animal* vs *Animals*). In Wust a synonym relation can be created between two tags, which means that posts tagged with one tag can be found under its synonym and vice versa.

Multiple inheritance and synonyms of tags contribute to platform quality. First, information can be found by searching for a tag. Discussions tagged with synonyms or inherited tags are also shown. Second, it helps to prevent duplicate discussions, because users actually find the discussions they wanted to start.

3.5. Tags are Contexts

In traditional systems, posts belong to exactly one section of the page. In internet forums there are threads categorized in subforums. In Reddit there are subreddits and Stack Overflow itself is a site of the Stack Exchange Network.

In Wust we make no distinction between tags and community-contexts. They are the same. A post belongs to a community if it is tagged with the respective context-tag. And as one post can have multiple tags it can appear in multiple contexts. This contributes to platform quality. Discussions are not started multiple times because they can exist in different sections of the site at the same time. There is only one which contains all responses in one place.

3.6. Views are Downvotes

We expected that users only upvote regularly and downvoting happens only in rare special cases (section 5.1). That is why we decided to only implement upvotes and count the unique views of users who did not vote as downvotes.

To make the quality score comparable for sorting we use *Dirichlet Prior Smoothing* introduced by [Zha+11]. It is a smoothed proportion of up and downvotes which takes the common up- downvote ratio into account. This makes it possible to compare post quality and sort responses to a post, as well as sorting all posts in a specific context. The exact formula and constants we use can be found in the implementation section 4.

The motivation of counting unique views as downvotes was to address the rich-get-richer problem and avoid mixing popularity and quality. Popularity and quality are two different quantities. Popularity is the amount of users who viewed a post, while quality describes if a post is worth reading.

When a post has better quality, more users upvote it, which lets it be shown higher in a list of posts. This results in more users viewing the post and it is becoming more popular. But because views are counted as downvotes, more popularity without more upvotes is pulling the post down in the list. This should have the effect that high quality posts get more popular without affecting quality. This needs to be evaluated in future work. Of course this approach only works trivially when a user has to click a post to read and vote it. When several posts can be seen at the same time, more investigation needs to be done to identify the viewed posts.

Making post quality more independent from popularity contributes to platform quality.

3.7. Relative post Quality

It is not possible to vote on a post itself. Posts only have a quality in respect to a context or another post: A relative quality.

This makes sense, since a post can have a high quality in one context, but a lower one in another. The same holds for responses. An idea connected to two problems can work pretty well for one problem, but is more a workaround for the other one.

This concept contributes to platform quality. Posts can be discovered in their context by their quality in that context. A post cannot dominate one context, just because it has a very high quality in another one.

3.8. Community Moderation

Every user can notice something wrong about a post of someone else. It can be spam, vandalism, mobbing, trolling, wrong information, illegal information or wrong punctuation. This can be distracting when reading, can spread wrong information and in the end is simply time consuming for everybody reading it.

In Wust we let all users vote on every change made to any post. Users who proposed improving changes should get more power in future change decisions. The exact numbers and formulas can be found in the implementation part in section 4.4. The different states of a change in the moderation system is illustrated in figure 5.

Users collect karma points per context. When one user edits the post of another user, a change request is created. Other users can vote for the change request to be approved or rejected. The number of votes required depends on the popularity of the post. The higher the popularity the more votes are required to approve or reject a change. When a request is approved, the editor's karma in the context of the post increases (based on post popularity). If it is rejected, the user's karma in the context decreases (also based on post popularity).

An edit can consist of changes to the title, the description, the contexts, the classifications and deletion of the post. Change requests for connections are not yet implemented. So all changes to relations (classifications and deletion) are instantly applied.

The changes that need more votes can be viewed in the moderation-section of the site. Users see one change at a time and can vote on them using 3 different buttons: *Doesn't make sense* to reject and revert, *I Don't know* to skip and *Makes sense* to approve and

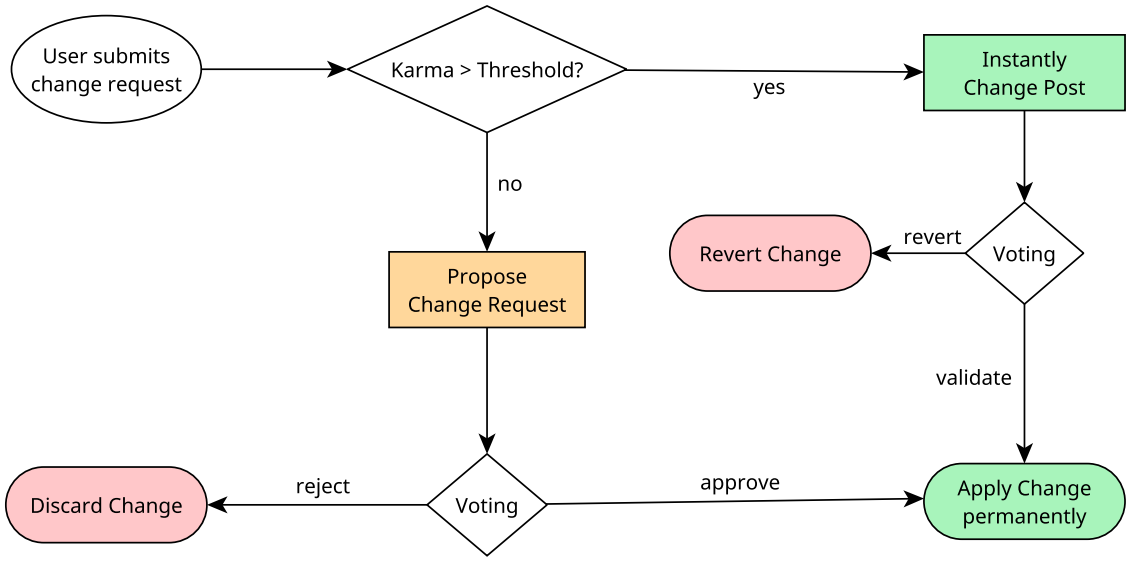


Figure 5: If the user of the submitted change has enough karma, the change gets applied instantly and other users can vote to validate or revert it. Otherwise the change becomes a request and users can approve or reject it.

validate. After clicking one of the buttons, the next change will be displayed (figure 6).

If a user submitted many approved change requests and therefore earned a lot of karma in a context, at some point, the user can make changes which are immediately applied to the post. In this case others can vote on the past change and validate or revert it. If the change is validated the editor's karma increases, if it is rejected the karma decreases and the change is reverted.

The karma threshold to edit a post is based on its popularity to make the system scale. It is usable in small groups working together. In this case most of the changes only need to be validated and the workflow is not interrupted by waiting for someone to approve a change. In large groups some consensus is needed, especially for popular posts.

Using the popularity as a base for the threshold motivates users to make higher quality changes which will be approved by many users. On one hand, the earned karma is exactly the threshold of the post. On the other hand, it is risky to make low quality changes to popular posts. On rejection the editor's karma decreases accordingly.

Users with higher karma have more vote-weight. This allows them to surpass the edit-threshold or vote with a higher weight.

The author of a post gets an initial boost, which, right after posting, lifts him above the edit-threshold. This allows to correct any mistakes immediately without approval

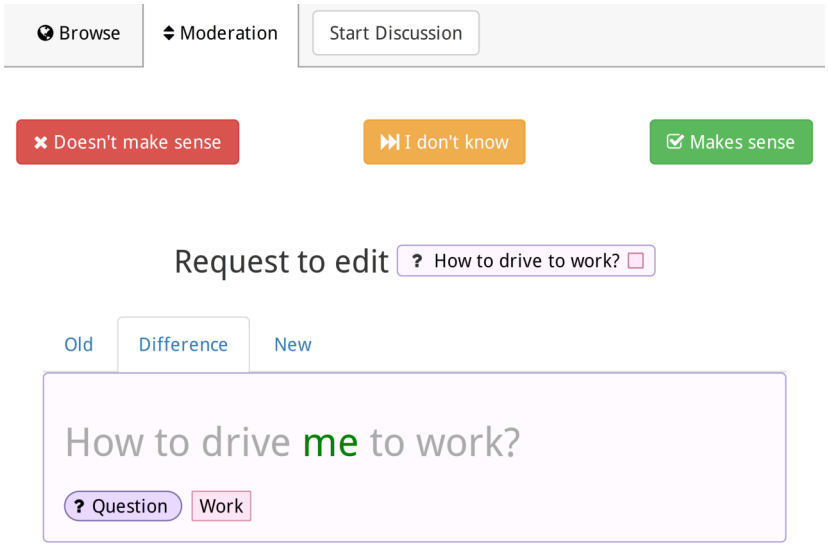


Figure 6: In the moderation view, users can decide what should happen to the presented change: *Doesn't make sense* to reject and revert, *I Don't know* to skip and *Makes sense* to approve and validate.

of other users. If the post gets very popular, the boost may not be enough anymore to surpass the edit threshold. In this case the post became a common good and even author changes need approval.

It is important to note that posts in a discussion can be reconnected and therefore reorganized. When done correctly this can eliminate redundancy and make the discussion represent its own summary. Assuming that users are intrinsically motivated to correct and reconnect

3.9. Live Updates

When interacting with the platform it is important to inform the user of new content created by other users. This is helpful to watch multiple users discussing without manually reloading the page. It also reduces the probability for creating redundancy. Cases where the user creates content or connections on the platform but does not know that something similar has been done already, should in general not occur anymore.

This contributes to platform quality, because new information is discovered efficiently. Users do not need to press a refresh button to check for new changes (polling). Instead they get the latest updates when they happen (event based).

3.10. Post and Tag Coloring

Every context-tag created on Wust is assigned a pseudorandom color based on the hash of its string. The hash is mapped to the hue dimension of the *HCL*[12] color space while the chroma and luminance are kept constant. This ensures an equal perceived brightness and saturation across all different tag-colors. The classification-tags have fixed hue values and have more saturated colors (figure 4). The fixed color values of classifications can bee seen in figure 7.

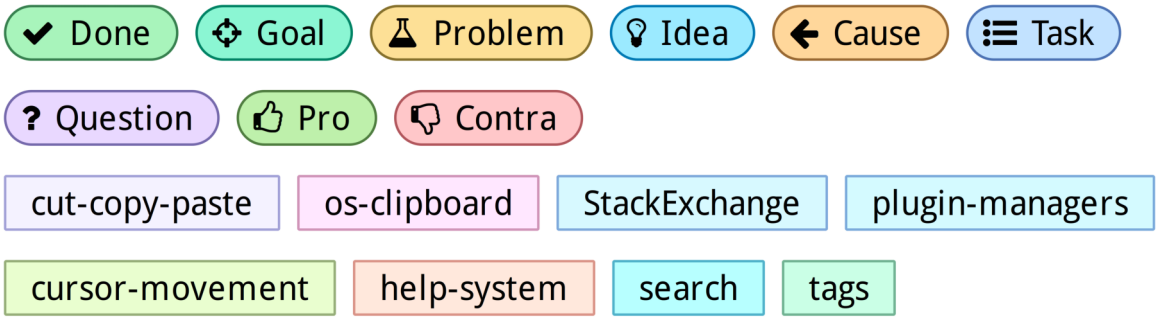


Figure 7: The classification-tags have rounded corners and symbols (top). The context-tags are brighter displayed as rectangles (bottom). Both have the same perceived brightness, which is achieved by using the HCL color model.

As the class of posts is implicitly defined by its outgoing relations, the color is also defined by the classifications of the relations. This gives colors in the system a meaning. To be consistent we almost completely eliminated colors from the remaining parts of the design. Posts without a classification color are displayed with a white background to communicate an "incomplete" or "unfinished" looking state and motivate users to applying a missing tag.

[12]The HCL color space is a cylindrical mapping for the *Lab** color space. The Lab* color space is designed, so that the euclidean distance in the color space represents the perceived difference of two colors.

4. Implementation

The previous chapter describes the concepts of the Wust discussion system. In this chapter we describe the general architecture and look at different interesting implementation details. We talk about the database abstraction developed for our hypergraph discussion model, details about the community moderation and the graph visualization.

4.1. Architecture

Wust is a Single Page Application (SPA). This means that the website never reloads, it fetches and sends data asynchronously. The backend only serves some static content, such as HTML templates, JavaScript, CSS and images and provides a REST API for data retrieval and manipulation with JSON.

The backend is implemented in the Scala programming language using the Play Framework. It is using the neo4j graph database for storage and manipulation of the hypergraph discussion model.

The frontend is written in EcmaScript 6 (ES6), the latest JavaScript standard, using AngularJS and the Bootstrap UI Framework. StyleSheets are written using SASS and compass for cross-browser compatibility and compiled to CSS. Some library functionality is written in ScalaJS, which is Scala compiled to JavaScript.

The graph visualization is done using the D3 Library [BOH11].

4.2. Database Selection

When selecting the database we had the option to use a traditional relational database or something newer from the NoSQL Family.

The first experiments with manipulating graph data in relational databases quickly lead to the conclusion that representing graphs with tables is not flexible enough for us. Every edge between two vertices is a N-to-N relation and therefore becomes a separate table for each pair of possible vertex types. Representing the concept that a post can be connected to another post or a connection resulted in very complex SQL queries. Graph traversal itself was difficult in this case. Querying a connected component for a post resulted in a big, complex query with lots of joins. So we looked for alternatives.

When considering databases from the NoSQL family, most were not fitting our requirements. These databases are designed to scale horizontally which is why they have to make compromises in the CAP theorem. Therefore, at this point in time, most of them did not feature transactions or alternatives. The massive speed and scalability comes from simplicity. As a consequence, there is usually no join support on the server side. This does not fit graph structures at all, because when traversing a graph one needs to jump from one vertex to other over edges. Doing this on the client side would lead to many requests to the database server and result in very bad performance.

Finally, the graph database neo4j seemed to be what we were looking for. It supports ACID transactions and a query language called Cypher which is well suited for graph traversals. Neo4j implements the property-graph-model [13], which means that nodes have different labels, relations have exactly one type and both have a set of key-value pairs attached. There is no fixed schema on the data, so modeling our hypergraph discussion model was straight forward.

4.3. Graph Database Abstraction

Working with the property graph model over the neo4j REST API requires interpreting the resulting JSON as graph structures. We implemented the library *renesca* to automate the HTTP communication and parsing of the JSON results. Since we had a fixed database schema in mind we created another abstraction layer called *renesca-magic* which generates boilerplate for type safety and property value casting.

4.3.1. Renesca

Neo4j can be used as an embedded database or over the network via a REST API. In the embedded case the data is stored in the file system and is accessed by using neo4j as a library. This allows to work imperatively with the complete graph and traverse and modify nodes and relations or to declaratively query data with the Cypher query language. The REST API provides access to nodes and relations via REST calls or Cypher queries.

Traditional relational databases have table data structures and the queries always result in a table form. Query results coming from the neo4j REST API can be graphs and tables. The existing Scala Neo4j REST libraries we tried were imitating *ORMs* (Object Relational Mapping) and were therefore limited to list or table structures of nodes

[13] http://neo4j.com/developer/graph-database/#property-graph

and relations. This was not convenient to work with, especially with our hypergraph discourse data structure. This is why we implemented our own open source library called renesca based on the following principles:

Work with Graphs instead of Lists Like the embedded version of neo4j, renesca allows to query a subgraph from the database and get the result as a graph or table data structure. The graph can be traversed like a Scala collection and properties are represented as hashmaps on nodes and relations. The property values can be casted to the expected type. The graph consists of three classes: *Node*, *Relation(startNode, endNode)* and *Graph(nodes, relations)*.

Track changes, persist later When modifying, creating and deleting nodes and connecting them with relationships it would be very expensive to submit a REST request for each change. In renesca we track changes and apply all of them at once when persisting the whole graph. This takes fewer REST requests and leaves room for optimization. Changes to properties are also tracked and persisted. This approach allows to pass around the graph structure in the code and persist once after all changes have been applied.

No lazy loading There is no further database traversing from a subgraph because there is no need to do so. When working with subgraphs retrieved from a query, we know which data we need in the future and can fetch that with the query before traversing. This approach saves a lot of unnecessary requests.

4.3.2. Renesca-magic

When working with renesca it was natural to write wrappers for specific nodes such as posts or tags. Posts always have a summary. So instead of looking up in the property hashmap and casting every time, we wrote a wrapper class which implements typed getters and setters for the needed properties. Posts also always have an author, so we can wrap the graph traversal with an appropriate getter. We extended renesca with simple schema helpers to wrap nodes, relations and graphs.

While the model evolved, it became very error prone to change the model and therefore refactor the boilerplate. At one point we decided to try to generate the boilerplate with Scala macros from a high level Scala DSL (Domain Specific Language). We succeeded and as a result we were able to reduce the model code by a factor of 10.

Scala macros are hygienic macros and therefore work on abstract syntax trees instead of strings of code. The trees read from the schema definition are transformed, typechecked and directly compiled. Additionally, a file containing code generated from the tree is produced. This helps with debugging and understanding the tree transformations applied.

We factored out the macro code as a library called renesca-magic. With the code generation infrastructure it was possible to implement more high-level abstractions which helped us in the development of the Wust discussion system.

Labels The names of the node and relation definitions are directly translated to labels and relation types of the property graph model.

Properties Renesca-magic generates wrapper classes and factories for Nodes and Relations. Both can have properties which can be primitives or optional primitives. The properties can be immutable by writing them with a *val* and mutable by using a *var*. Default values can be specified with an assignment of an expression which is evaluated on creation. The classes are generated with getters and setters, taking mutability and optionality into account. The factories provide methods to wrap existing nodes or relations and methods to create new instances with the required properties and the optional ones as default parameters.

Graphs There is a wrapper for the whole graph, which provides access to the different types of nodes and relations contained in the graph. This graph can be persisted like the graph from renesca.

Relations and Neighbors The wrapper for relations takes two additional parameters. The start node and the end node of the directed relation. This triggers the generation of accessors in the start node and end node wrappers to access neighbors over this relation in both directions.

Hyperrelations Hyperedges in mathematical terms are edges which connect an arbitrary set of vertices. In renesca-magic we define Hyprerelations as relations which get all characteristics of a node. They can be used as a drop-in replacement for nodes and relations. hyperrelations can therefore connect two nodes and be connected with other nodes. So this is a specialized form of the mathematical definition. Internally

in the generated code they are represented by a node with an incoming and outgoing relation.

Multiple Inheritance When using the same property over and over again on different types of nodes it makes sense to define it only once in a trait and compose it into all the needed nodes by inheritance. This helps to keep the schema definition DRY (Don't repeat yourself). The name of the trait is added to the list of labels. As in object oriented programming, all children of the trait can be handled as the same type. This allows to work with collections of nodes sharing the same properties. There are also traits for relations with the same functionality.

Renesca-magic has other features such as index annotations and lazy match and merge constructs. On top of this in Wust we implemented a DSL (Domain Specific Language) which declaratively maps routes onto parts of the database model. More on that can be found in the work of Karoff [Kar15].

4.4. Community Moderation

The community moderation system was designed to scale. It should work with few users as well as with a large number of users. Small groups should not get stuck with having not enough vote-weight to accept changes. While large groups should not need too many votes for changes to get applied.

The moderation system is implemented by distributing *karma* points to users. A user has karma per context, which indicates how many good actions a user made in this context. The karma gives the user a higher vote weight for change decisions.

The vote weight v_{uc} per user u in context c is defined as the integer base-2 logarithm of the users karma k_{uc}, but at least 1:

$$v_{uc} = \lfloor log_2(\max(k_{uc}, 2)) \rfloor$$

The weight is defined to give users more power when they have made good actions in the past, but prevents the best editors to form an oligarchy. The vote weight increases when the earned karma is an order of magnitude higher. This is similar as in the WikiTrust quality system.

Every post has an edit threshold. It defines how many votes are needed to permanently apply a change (approve or validate). The edit threshold e_i of the post i is defined by the square root of the popularity (unique view count) p_i of the post plus 1:

$$e_i = \lfloor \sqrt{p_i} \rfloor + 1$$

We chose the square root of the number of views, because it gave good perceived thresholds in small and big groups. For example, when two users are working together, both have to approve changes of each other. When ten are working together, 4 have to approve changes. When 100 are working together, it needs ten approvers. A linear threshold turned out to be too low with small groups and too high in big groups in our test environments. A Logarithmic threshold was way too low in big groups. This is definitely a formula that can be improved in the future.

The reject threshold r_i is defined as the half of the negative edit threshold but lower or equal to -1:

$$r_i = \min\left(-\left\lfloor \frac{e_i}{2} \right\rfloor, -1\right)$$

This takes into account that users do not downvote regularly and the assumption is that rejections should happen much faster than approvals, to not waste time on changes that do not improve the content of the post.

When editing a foreign post, there are at least two unique post views. The one from the author and the other from the editor. This results in an edit threshold of at least 2. It means that at least 2 users with a vote weight of at least 1 need to vote on an edit of a new post to go through.

When users vote on a change j for post i, the result s_j is defined as the sum of all weighted upvotes up_j minus the sum of all weighted downvotes $down_j$. When this sum reaches the upper edit-threshold e_i or lower reject-threshold r_i, the change is approved/validated or rejected/reverted respectively:

$$s_j = up_j - down_j$$

The author of a post gets a boost of 33 on his vote-weight of that post. This means that $(33 - 1)^2 = 1024$ the author can do instant validated changes until 1024 other users have seen the post. With such a high popularity the post becomes common good

of the community. Further on the author needs the approval of other users or a higher vote-weight in that context.

When a change gets approved or validated, the editor receives karma in the size of the edit threshold. In contrast, when a change is rejected or reverted, the editor looses karma in the size of the edit threshold.

A change affecting text and one or multiple tags is automatically split into atomic change requests describing only one change. Changes to title and description belong together and are one atomic change. Context-tagging and classifying are another atomic change. This allows to do some simple merging of duplicate changes.

For text changes, the old and the new version of the text are stored in the change. This allows to calculate and display a text difference view. A text change can only be applied, if the old version equals the current version of the post. When two users attempt to change the text of the same post, one of the changes gets applied first. The second change cannot be applied, because the old version of the change does not equal the new current version of the post. Therefore, the change is marked as conflict and discarded. In the future, to avoid most conflicts, a patch representing the change could be generated. This patch can be applied on top of the other change if they are not changing the same sections. The conflict is resolved automatically. Else there is a merge conflict which needs to be resolved by user intervention.

Currently it is not possible to pull back a change request or instant change. It could make sense to allow this when nobody else has voted on the change yet.

Vote-Stream All change requests are already displayed below a post in the focus view where users can vote on them (6). At the same time change requests and unvalidated changes are displayed in the vote-stream. Users see one change at a time. The change is described with a sentence and details are displayed depending on the type of change. The goal is to have a very simple interface with the lowest possible barrier to motivate users to vote on others changes.

- **Title and Description:** Changes in the text of the title or description are displayed in a tabbed view. On the left side is the old version, on the right side is the new version. The middle tab is labeled with *Difference* and shows a word-based diff. This means that the deleted words are stroked and colored red and the added words are colored green. The words which stayed untouched are displayed with gray color.

- **Contexts and Classifications:** When adding tags, the change is indicated with the sentence: *Request to tag <Post Summary> with <Context> as <Classification>*. The last part with the classification is only displayed if a classification was added. When removing tags, the sentence is: *Request to remove <Context> as <Classification> from <Post Summary>*. Additionally, the whole post with description and its other tags is displayed. The tags involved in the change are highlighted.
- **Deleted posts:** Deleted posts are displayed with a red border.

Users can decide what they think about the currently displayed change with 3 buttons. After pressing one button, the next change is displayed.

- **Doesn't make sense:** When pressing this button, the change receives a downvote.
- **I don't know:** This button skips the current change and saves that the user skipped the change, so it will not be displayed again.
- **Makes sense:** When pressing this button, the change receives an upvote.

Reputation system classification The voting system on change requests and past changes describes a community moderation system which addresses some drawbacks of existing systems. It allows everybody to criticize established moderators and qualitatively decrease their power when necessary. The moderation ability is expressed in karma points which make the moderation system a *reputation system*. The system can be classified by the taxonomy of [HBC15] as an **explicit** reputation system with the following dimensions:

- **History: Global** The karma points are stored centrally on the server instead of on the client side.
- **Context: Multiple** Karma is defined per user per Context-Tag, instead of only per user.
- **Collection: Indirect** Karma points are distributed after a change was validated, approved, rejected or reverted by a group of individuals who were not involved in the change themselves.
- **Representation: Discrete** Karma points are represented as integers.

- **Aggregation: Counting** The karma of a user is the sum of all gained and lost karma points.
- **Entities: Individual** Karma is clearly assigned to individual users.
- **Presence: Online** Karma information is continuously present and available from the server.
- **Governance: Centralized** The server takes all decisions of how the karma points are distributed.
- **Fabric: -** This dimension is not defined for our reputation system.
- **Interoperability: Open** Karma for a user in a context can be obtained using the REST API.
- **Control: Incentives/Discentives** Users are motivated to gain karma by doing good changes in the system.
- **Evaluation: Holistic & Atomistic** The vote weight is decided on all the karma a user collected in a context. The history of karma gaining / losing is shown on the users profile page.
- **Data Filtering: None**
- **Data Aging: None**

4.5. Post Quality Estimation

Users can vote on a post in two different ways: (1) with respect to a context and (2) with respect to the post it is responding to. Both votings try to estimate the quality of a post. This estimation is used to sort the posts when they are displayed with other posts. All upvotes $n_\uparrow$ are counted. Unique views of users who did not vote are counted as downvotes $n_\downarrow$:

$$n_\downarrow = \text{views} - n_\uparrow$$

We use the *Dirichlet Prior Smoothing* technique to calculate a score $s(n_\uparrow, n_\downarrow)$ for $n_\uparrow$ upvotes and $n_\downarrow$ downvotes:

$$s(n_\uparrow, n_\downarrow) = \frac{n_\uparrow + \mu p_\uparrow}{n_\uparrow + n_\downarrow + \mu}$$

This score is basically the proportion of upvotes in the number of votes, but has some advantages over the trivial approach. The parameter $p_\uparrow$ is the probability of a vote being an upvote. It can be approximated by all the votes already seen in the system. The constant $\mu > 0$ describes the smoothing of the score. It expresses how strong the prediction is taken into account when only few votes are available. This makes it possible to compare posts with few votes to posts with lots of votes. With this approach the score of posts without any vote is exactly defined.

In our case, since we had no experimental data of our system yet, we expected the upvote behavior to be less active than the viewing behavior. So we set the probability of an upvote to $p_\uparrow = 0.1$. We set the smoothing to $\mu = 10$.

4.6. Graph Visualization

The graph view shows a visualization of the discourse hypergraph. It is implemented using the force directed graph feature of the D3 JavaScript Visualization library. D3 stands for *Data-Driven Documents* and is a visualization library for almost any kind of data. It is very flexible, because it is a helper library for generating SVG and HTML code providing some additional high level features, such as force simulations, color calculations, scales visualizations, and different kinds of diagrams.

Force Layout The force layout of D3 is a particle simulation implemented using *position Verlet integration*. This means that each particle stores its position x and its previous position x^*. This implicitly defines the velocity as $x - x^*$ and the next position is calculated by $x + (x - x^*) = 2x - x^*$. To apply forces to a particle only the current position needs to be moved. This makes it possible to define a complex set of forces accenting the particle. All forces, such as pseudo-gravity, repulsive charge and fixed link distance of graphs are applied like that. A convergence is achieved by multiplying the forces with a cool down parameter $\alpha = 0.1$ which is decreasing (multiplied by 0.99) with every time step until $\alpha < 0.005$. At this point alpha is set to 0 and the simulation is stopped.

HTML and SVG Layers The force simulation only calculates the particle positions. To visualize them HTML or SVG code needs to be generated. In Wust every post in a graph becomes a particle and is rendered with HTML. It displays the summary and little colored squares, indicating the contexts. Rendering posts with HTML gives us the advantage of reusing CSS defined for other posts on the website. It also allows the

HTML rectangles to have a size dependent on its contained text. This is not possible with SVG.

Since we cannot draw arbitrary lines with arrows in HTML, we created an SVG layer below the HTML layer with the exact same size and coordinate system as the HTML one. Here we can draw the relations as SVG paths with markers as arrows. The result is a beautifully looking graph which has the advantages of HTML and SVG.

Deterministic Convergence Result When the force layout is initialized, all nodes are positioned randomly on the area. This gives a different convergence result every time. To get consistent results we hashed the *ids* of the nodes to get pseudo random positions. This always produces the same convergence per graph. In the future, it could be improved to produce similar convergences for similar graphs.

Hyperrelations Hyperrelations in the graph data are normal nodes with a flag to indicate that they are hyperrelations. hyperrelations consist of a node and two relations, representing a relation which can be connected to other nodes. Only the outgoing relation gets an arrow to look like one relation with a label. When running the simulation and looking at the result, the hyperrelations behave like normal nodes and most of the time the two relations are pointing from the node in two different directions (figure 8). The unit of one node and two relations is difficult to recognize as one assembled relation.

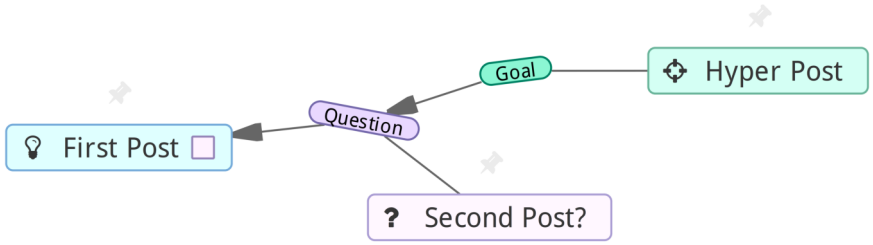


Figure 8: Hyperrelations consist of one node and two relations. The node *Question* behaves like all the other nodes in the force layout and does not lie on a line between its incident nodes *First Post* and *Second Post?*.

To overcome this, we created a force f_{hi} which pushes the node position x_i towards the center c_i of its start node x_{is} and end node x_{ie}:

$$c_{se} = \frac{x_s + x_e}{2}$$
$$f_h = (c_{se} - x) * \alpha$$
$$x_i \leftarrow x_i + f_{hi}$$

With this force the nodes are lined up and hyperrelations are easily recognizable as such (figure 9).

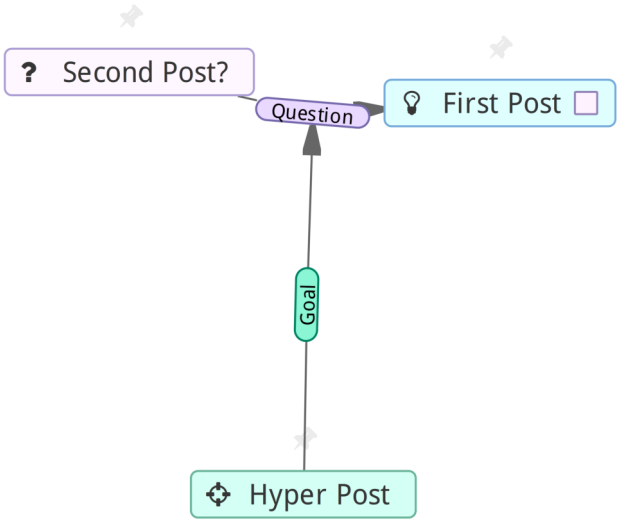


Figure 9: The hyperrelation *Question* is pushed towards the center between *First Post* and *Second Post?*, and is recognizable as a hyperrelation. It can be seen that the displayed relation lines are very short.

Consistent Reading Direction The focus view displays the successors of the focused post on the left and the predecessors on the right. To keep a consistent reading direction with the focus view, we need to create forces which pull successors to the left and predecessors to the right.

For each node, we calculate the count of *deep successors* and *deep predecessors*. Deep successors are all nodes reached by a depth first search from the current node and iterating only over outgoing relations. Deep predecessors are analogously the nodes reached by iterating over only the incoming relations. We calculate a force-weight r_i for each node as the difference of the inverse squares of deep successors suc_i and deep predecessors pre_i:

$$r_i = \frac{1}{\text{pre}_i^2 + 1} - \frac{1}{\text{suc}_i^2 + 1}$$

The weights are shifted, so that all positive and negative weights sum up to 0 and create a force equilibrium where n is the count of nodes:

$$r_i \leftarrow r_i - \frac{\sum_i r_i}{n}$$

and normalized, so that all weights lie in the interval $[-1, 1]$:

$$r_i \leftarrow \frac{r_i}{\max(|\max_i r_i|, |\min_i r_i|)}$$

The strength of the force is scaled by the square root of the number of nodes and applied to the current position:

$$f_{ri} = \begin{pmatrix} r_i * \sqrt{n} * 20 * \alpha \\ 0 \end{pmatrix}$$
$$x_i \leftarrow x_i + f_{ri}$$

This achieves the desired behavior and pulls the successors to the left and the predecessors to the right (figure 10).

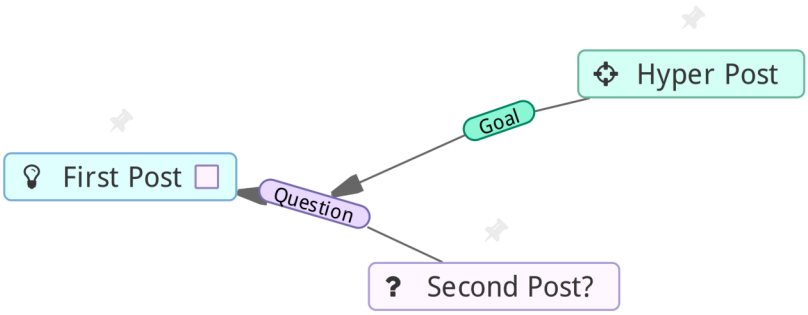


Figure 10: Posts with many predecessors are pulled to the left and posts with many successors are pulled to the right. This makes the reading direction consistent with the focus view. It can be seen that some relation lines are almost too short to be visible at all. The lines are drawn between the centers of the nodes, which are pushed to have a constant distance.

Constant visible relation line length The *linkDistance* constraint of D3 works by doing a *Gauss-Seidel relaxation* on the distance of the link. This has the effect of pushing close nodes away and far nodes closer to each other.

When visualizing arbitrary graphs, the nodes are usually drawn as circles. In D3, the lines for displaying relations are connected between the centers of nodes. The filled node circles overlay the lines, so that the segment inside the circle is hidden. Since the distance from the center of a circle to its outline is constant in all directions, a constant line distance between two node centers also produces a constant visible line. Displaying arrows is also trivial in this case, because the arrows can be displayed at a constant offset from the line endpoint.

Post nodes in Wust contain text and are displayed as rectangles. For a rectangle the distance to its outline is not constant in all directions. Therefore, the displayed line length between two rectangles which have a constant distance between its centers is not constant for every rectangle size, rotation and position constellation. This results in unequal line lengths between rectangle nodes in a graph. It also makes it difficult to display arrows correctly.

To overcome this effect, we calculate the visible line segment and draw that alone. To calculate this segment we intersect the whole line with the rectangles at its endpoints and cut away the hidden parts. This intersection was implemented for arbitrary convex polygons, because this also covers the rotated rectangles, used for the classification labels on the connections. This already solves the arrow drawing issue. Also, we do the Gauss-Seidel relaxation on the displayed line instead of the whole line. This achieves a constant displayed line length shown in figure 11. For connected overlapping nodes the visible line is not defined. In this case we feed the relaxation with a negative distance depending on the overlap of the nodes to push them away from each other.

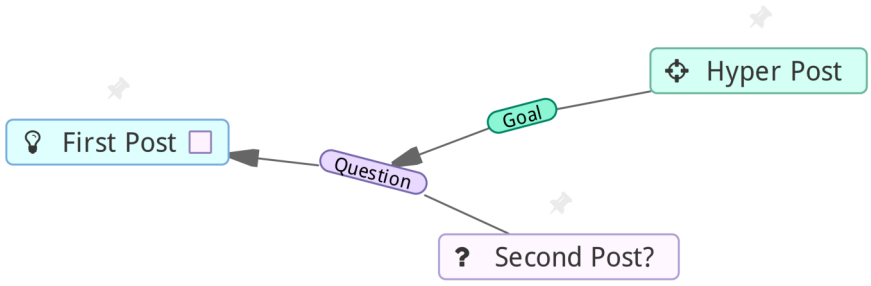


Figure 11: Hyperrelations are aligned, the reading direction is consistent and the visible relation lines are pushed to have constant length.

In some cases the forces are not strong enough to push the nodes away from each other. It can happen that two or more nodes are overlapping. This happens in graphs with a node of high degree or a high number of nodes. It could be solved in the future by implementing collision detection or a strong repelling force for very close nodes.

5. Evaluation

The concepts and design decisions of Wust are based on several hypotheses that we try to evaluate with two user tests and one questionnaire. In our user tests we also had several insights which helped to improve our concepts and already confirmed some of our hypotheses.

5.1. Hypotheses

In this work we have six hypotheses about the Wust discussion system and its concepts. The first one **H1** is about system usability in general using the system usability scale [Bro96] and split into several sub-hypotheses (**H1a-e**). The other five hypotheses (**H2-H6**) are about moderation and voting. Karoff [Kar15] shares the usability hypotheses with this work and evaluates others in regard to discourse structure and overview.

5.1.1. Usability

H1: Users find the system usable. When designing a discussion system it should be usable for everyone. The more diverse the participants are, the more diverse are the perspectives on one topic and so are the insights and outcomes of the discussion. With diverse users we mean users with different knowledge, different age or sex.

H1a: Users find the system usable, regardless of their familiarity with graph theory. There was a claim that only people with knowledge in graph theory can understand and use a discussion system based on hypergraphs. Our hypothesis holds against this. We assume that not the concept but the terminology of graph theory is a problem. Users can also understand the graph structure of mindmaps and social networks, so they should be able to understand a hypergraph based discussion system.

H1b: Users find the system usable, regardless of their self-efficacy using technology.

H1c: Users find the system usable, regardless of prior experiences with collaborative online platforms. The ideas of Wust adapt and build on several concepts already present in other online platforms like Reddit and Stack Overflow. Still, users who are not familiar with these platforms should be able to quickly understand the concepts and learn how to use the system. We assume that the ability to learn heavily depends on how the concepts are presented and introduced, instead of their complexity. Therefore, with the sufficient learning material and introduction, users should be able to learn Wust, regardless of their prior experiences with online platforms.

H1d: Users find the system usable, regardless of their sex.

H1e: Users find the system usable, regardless of their age.

5.1.2. Moderation, Voting and Ranking

H2: Users are intrinsically motivated to correct incorrect information. A Moderation system can only work when users correct the wrong information they encounter. The motivation to do this can be intrinsic or extrinsic. We expect that there is an intrinsic motivation to correct punctuation, wrong information and wrong connections. This hypothesis supports the choice of implementing a community moderation system. When there is a fixed proportion of users who can handle the amount of content everybody is producing, the moderation system scales.

H3: Users agree that empowered users are allowed to edit and reconnect posts. The concept of users editing the content of others can only work when the authors agree that their work can be modified. Else it could be possible that they get upset and leave the platform.

H4: A collaborative moderation system works by users voting on the changes made by others. In Wust, users can vote on changes made by other users before or after they get permanently applied. We expect that this approach makes sense to users. We further expect that users think that users who did good changes in the past are the right ones to get more moderation permissions.

H5: Users only vote on positively outstanding posts. Based on this hypothesis, Wust only implements upvotes and interprets the unique views of users who did not vote as downvotes.

H6: Users associate upvoting with quality and content discovery. On Reddit and Stack Overflow content is discovered by looking at lists of highest voted questions and start posts respectively. The assumption is that users realize that even when this list is not visible on the screen, the voting button's main purpose is to affect its position in these lists. In Wust a post has vote buttons for every successor[14] and every Context-Tag. The motivation behind this is that posts appear in different lists where they are compared to posts which reply to the same post or are tagged with the same Context-Tag. One post can have higher quality in one list while having lower quality in another. For example a post can have high quality in *Mathematics* but low quality in *Politics* and vice versa. Users should be able to discover high quality posts in these lists and therefore be able to upvote a post for these specific lists.

5.2. Method

The hypotheses are evaluated using two user tests and one questionnaire. The first user test's intention was to find the primary user interface (UI) difficulties and inconsistencies for browsing, responding to and modifying a graph based discussion (section 5.2.1). The second user test evaluates the fixes and concepts which emerged after the first user test and further focuses on hyperedges and the moderation system (section 5.2.2). After some more fixes to the UI and the discussion model, a broader audience was confronted with the system and an introduction tutorial. They had to learn the concepts of Wust themselves and afterwards answer a questionnaire about usability, discussion structure and moderation. The collected data is used to evaluate our hypotheses quantitatively (section 5.2.3).

5.2.1. User Test 1

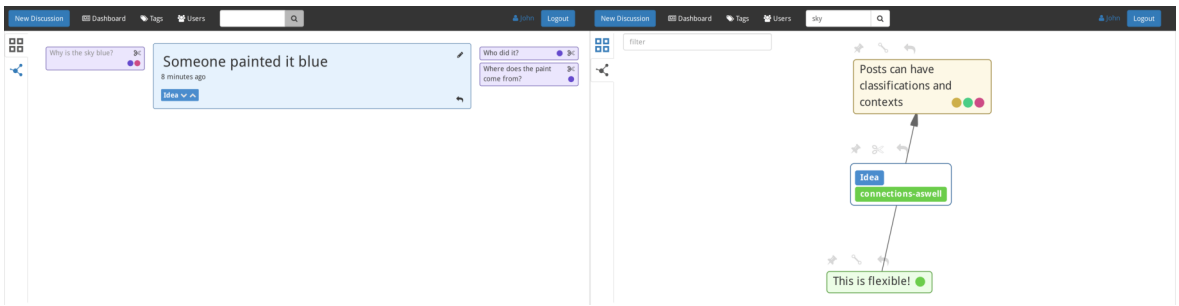
The goal of the first user test was to see if users can understand discussions in a graph structure. They should use and test the main graph interaction tools. This includes navigating in the focus view and having one post responding to two other posts. It

[14] A successor as known from graph theory. Predecessors of a node represent its responses and successors of a node represent the nodes it is responding to.

should also help to identify the biggest UI-problems that were preventing the subjects from learning the system on their own.

The user test consists of twelve tasks (appendix A). Every subject had to execute the printed tasks sitting at a desk with a laptop running the website and a wheel mouse. The whole process was supervised and the screen contents as well as the audio captured. Subjects were encouraged to speak out their thoughts while doing the tasks. After the session they were able to give feedback and talk about the system in general.

State of Development At that point in time the system looked and behaved quite differently than the system described in this work. Moderation was not implemented yet and every user was allowed to change everything. The discussion model allowed to apply context-tags as well as classification-tags to posts and connections. The intention was to provide separate classifications for posts and connections, where posts have nouns such as like "Idea" and connections verbs like "solves" or "repliesTo". There were no connection indicators in the focus view, only a scissors-symbol to disconnect. The button to respond had a reply symbol showing a curved arrow pointing to the left.



(a) Focus View with scissors (b) Tags on Posts and Connections

Figure 12: At the time of the first user test, Wust had scissors to disconnect in the Focus View (a) and allowed both types of tags on posts and connections (b).

Tasks The tasks started with the website showing one post in the focus view and the user was already logged in. The subjects had to read it and describe what else they see on the screen. Then they had to navigate to the focus view of an answer and back. One answer was completely unrelated to the discussion which they had to remove. Next, they should ask a specified question to the current post. Afterwards they should switch to a view with more overview (graph view) and describe what they saw. They had to respond with an idea to a problem, connect it to another existing problem and remove the just created connection. The last task was to start a new discussion and edit its description afterwards.

Subjects The first user test was done with four subjects. Most of them being students from mixed faculties.

Results The behavior of the subjects allowed us to reconsider the discussion model and many UI concepts. The key insights are:

- **Subjects understood a discussion represented as a graph.** When navigating through the posts in the focus view the users were a bit confused that the response were displayed at the right, but quickly understood that the left and the right side were the other posts connected to the current focused post. In the graph view they always completely understood how the posts are connected to each other. It also made sense to everybody that one idea can be suggested for two problems at the same time. Some immediately said that it reminded them of a mind map. Subjects suggested to display arrows between posts in the focus view, like they are displayed in the graph view.
- **Subjects thought that the tags in the focus view were buttons and clicked them.** The initially focused post was tagged as a question. One of the task was to ask a question. Almost everybody clicked the displayed "Question" tag. They said it looked like a button. And they were not able to find anything else that made more sense to them to ask a question. This let us redesign the tags with lighter, less intense background-colors and black text.
- **Subjects classified posts when creating them.** When starting a new discussion or responding to a post, the subjects intuitively selected one of the suggested classifications for the post themselves. This strengthens the concept of structured discussion systems with classified posts. It also motivated to display tag-suggestions in a prominent place below the title input field of the post.
- **Subjects used post-classifications for connections.** In one task subjects had to connect two posts in the graph view. After connecting, a pop-up dialog appears, asking for tags to apply to the connection. Since the tag-suggestion feature suggested a limited amount of tags, it suggested only classifications, while other tags would have been possible as well. But every subject naturally selected a classification to describe the connection. This behavior led to two conclusions for Wust. First, when suggesting tags, there should always be both classification tags and context tags. Second, change the discussion model to have classifications on the connections which implicitly define the classes of predecessor posts. This result also shows that subjects realized that the connection between posts has a semantic meaning for the two connected posts.

- **Subjects found the reply button confusing to ask something.** When subjects had to ask a question they were looking for a button to ask a question and then clicked the "Question"-tag as stated above. But they were not considering the button with the reply-symbol. Even after telling them, pressing of the button felt wrong and confusing. This led us to experiment with different symbols, like a comment-bubble and a plus. In the end it was the label "Respond" which worked quite well and describes the general action of responding to something said by someone else.
- **Subjects suggested that the graph view should be more organized.** The subjects stated that the graph view gives a good overview of how the posts are related to each other. But the positions do not have any meaning. Some posts appear on the top and some at the bottom. And most of the time the start post is positioned in the center with its responses around it. This motivated to imitate the order of the focus view where responses are on the right and the start post on the left.

The first user test confirmed that subjects can work with discussions represented as graphs. It also helped to identify the worst UI-problems and their solutions. The most important result is that it helped improve the graph discussion model to make it more intuitive and efficient. The next section describes how the second user test evaluated the system after all corrections were made and a community moderation system was implemented.

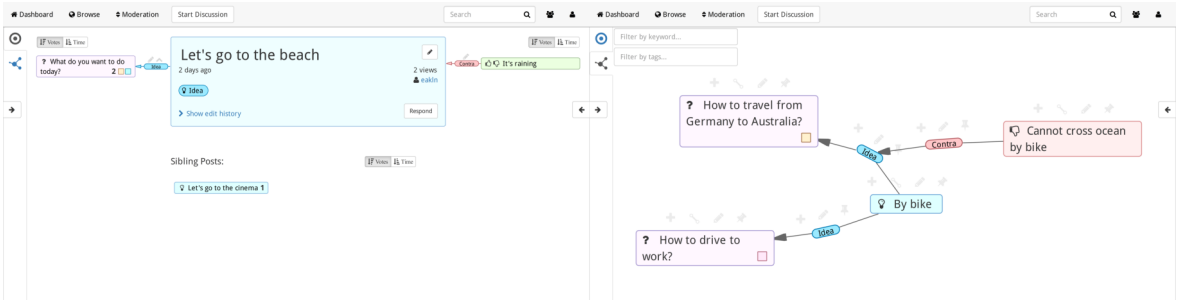
5.2.2. User Test 2

The first goal of the second user test was to verify the understanding of the graph representation and see if it extends to hyperrelations. The second goal was to introduce the concept of community moderation and find how users handle redundancy. It also evaluated the solutions for the UI-problems identified in the first user test.

This time the user test consisted of 19 tasks (appendix B). ten for the discussion graph representation, three for the concept of hyperrelations and six for community moderation. The setup was the same as in the first user test.

State of Development At the time of the second user test, Wust advanced in many directions. It had a brighter design and tag colors generated with the HCL color model. This makes them have the same perceived brightness. The discussion model is the final

one described in this work and now has classifications on the connections and context-tags on the posts. The graph view received many visual improvements as described in section 4.6. And the community moderation is completely implemented with users having karma per context.



(a) Focus View (b) Graph View

Figure 13: The state of development of the second user test has an improved design and UI, implements the new discussion model and community moderation.

Tasks The tasks of the second user test started with the website showing a post in the focus view. Similar to the first user test the subjects had to navigate through the responses two levels deep and back. They had to ask a question and then view the discussion in the graph view. This time they had to describe if the graph view presents a good overview. Again they had to add an idea, connect it to another post, remove it again and start and edit a new discussion. Then the subjects had to look at a discussion with two questions and one idea connecting them, like in figure 13 (b). They had to add a contra argument which only fits to the idea in relation to one question. For the other one it did not make any sense. This was a chance to discover the concept of connecting the argument to the connection between the idea and the respective question. Afterwards they had to edit a post from another user but had not enough karma for the change to get applied instantly. They were asked to explain why their change was not visible. To show the concept of changes of other users, they had to open the vote stream and rate two changes. Afterwards they had to remove a spam post from a discussion. The last task was about a question with another question asking for clarification. The subject had to edit and extend the original question to make the clarification question unnecessary. After that, they should decide themselves what to do with the redundant question.

Subjects For the second user test eleven subjects finished the test and gave their feedback. Many of them were students from different faculties, but also pupils and employees participated.

Results The results show that the UI-problems identified in the first user test are solved and that the graph view gives a good overview now. Subjects understand the concept of hyperrelations and can handle community moderation as well as redundancy. The key insights are:

- **The UI needs small improvements to make the new discussion model more intuitive to work with.** After responding with an idea to the initial post, the subjects had to connect the idea to another post. Roughly one third of them did not choose a classification for the connection and were further not confused about the new connection not having the label "Idea" on it. They expected the post itself to be an idea and that the connection should carry that respective classification automatically. In only very few cases one post has different classifications for outgoing relations. In the future the UI could pre-select the common classifications of the already existing connections. The user test suggests that this is the intuitive behavior.

- **Subjects understand the concept of hyperrelations.** When letting the subjects attach the argument, most of them attached it to the idea. Some naturally came to the conclusion that it makes most sense to attach the argument to the connection between idea and question. After explaining that this is possible, most subjects reconnected the argument. A few left it connected to the idea. Two mathematicians explained that the concept makes sense but is probably too complicated for ordinary people.

- **Subjects can handle community moderation with prior explanation.** During the test, the subjects were a bit confused why the change did not appear immediately and explained it as a software bug. The concept became clear when showing them the vote-stream with the changes of other users. Finally, everybody understood it after an additional explanation. The concept of karma points was only introduced to the subjects who were interested. Afterwards some subjects independently had the idea that karma should be assigned per context.

- **Half of the subjects eliminate redundancy.** When confronting the subjects with a clarification question that was already answered, subjects behaved differently. Roughly half of them marked the question with the classification-tag "Done". The other half deleted the question with the explanation that the question does not have a reason to exist anymore.

The second user test shows that the new graph based discussion model works and is more intuitive. But the UI needs to be adapted a bit more to reflect intuitive connection

behavior. Subjects need more hints to show that hyperrelations are possible in the system. The moderation system needs some explanation but eventually was intuitive to everyone. Redundancy was eliminated by roughly half of the subjects. In general the system proved to be usable and intuitive. In the next section we evaluate with a questionnaire if users with only a little explanation by a tutorial can learn the system and find it usable.

5.2.3. Questionnaire

To quantitatively evaluate our hypotheses, we prepared a questionnaire with 17 questions for subjects to answer (appendix C). The questionnaire starts with personal questions, like age, sex, online platform usage and graph theory knowledge. It continues with the system usability scale [Bro96], questions about the acceptance of a structured discussion model, arrow directions, hyperrelations and community moderation. The survey was taken by all subjects from the second user test and subjects who visited the Wust website and completed the tutorial.

Tutorial To explain the concepts of Wust as simple and efficiently as possible, we created a tutorial in the navigation bar that guides the subject through the different possible actions and explains the concepts of Wust. The last step in the tutorial links to the questionnaire. The tutorial expands when hovered with the mouse and collapses automatically when interacting with the website. This allows to get familiar with the system without the need to switch to another window to read about the next step.

There was only one server instance which all subjects were able to access simultaneously. This allowed the subjects to interact with each other on the platform.

The tutorial requires some discussions and change requests to be present. It tells subjects to ask a question to a foreign post and rate change requests of other users. To make sure that each new subject has the ability to go through the tutorial without problems, we designed the tutorial to let the subjects produce the required artifacts for the next subjects. Subjects are told to start new discussions and edit foreign posts. This resulted in fresh posts for the next subjects to ask questions and produced change requests to vote on. In the end it helped many subjects to get familiar with the system and therefore ready to answer the questions of the questionnaire.

(a) Collapsed tutorial (b) Finished tutorial with link to questionnaire

Figure 14: The tutorial is expanding when hovering with the mouse. It provides 8 tasks, gives explanations and finishes by linking to the questionnaire.

Questions The questions are designed to evaluate our hypotheses. To find out about usability hypotheses (**H1** and **H1a-e**), we used the questions from the system usability score and asked for age, sex, usage frequency of different online platforms, familiarity with graph theory concepts and self-efficacy using technology. We asked questions about discussion structuring, graph arrow reading directions and hyperrelations to evaluate the hypotheses of Karoff [Kar15]. The questionnaire finishes with questions about voting behavior, moderation, net-promoter-score [Rei03] and textual feedback.

Statistical Methods The collected data was used to evaluate user hypotheses statistically. T-tests were used and correlations were calculated. We chose $\alpha = 0.05$ as the significance level, which means that in 5% of the cases we find a significant effect or relationship, even if there is none. As sensitivity, $\beta = 0.20$ was chosen, i.e. in 20% of the cases we are unable to find a significant effect or relationship, even if it exists. The threshold for the minimal effect size of our sample was calculated using *G*Power*. All scales were evaluated using a 6-point Likert scale.

Our sample size consists of 51 subjects, including 15 females, 35 males and one unspecified sex. The average age of our subjects was 30.90 ($SD = 12.54$).

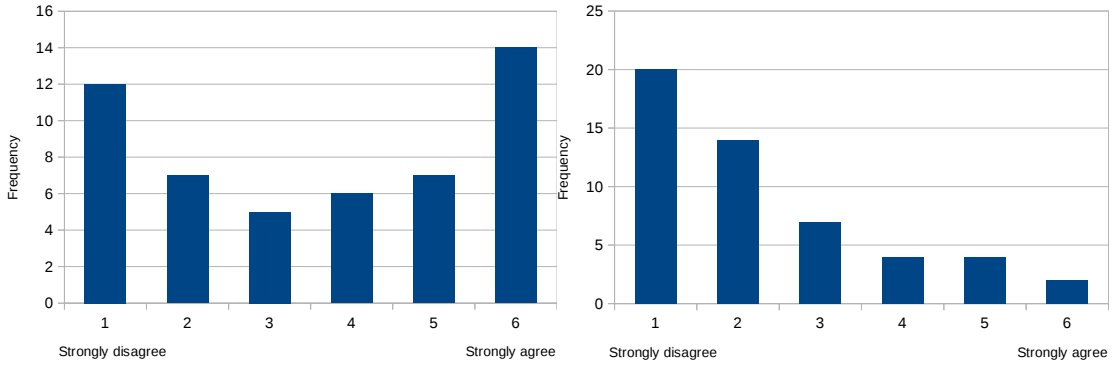
5.3. Results

We evaluate our hypotheses with results from the two user tests and the data collected from our questionnaire. The hypotheses are validated using qualitative and quantitative results.

The results form our user tests and questionnaire show that the perceived usability is high and does not correlate with graph knowledge, self-efficacy using technology, online platform experience, sex or age.

H1: Users find the system usable. In the user tests the feedback showed that the subjects found the system usable. We additionally confirm that with the system usability scale from our questionnaire. SUS is a 6-point Likert scale of ten items for measuring usability. For our sample the scale shows a good reliability ($Cronbach's \alpha = 0.867$). The mean SUS-score is 70.45 from 100 ($SD = 15.04$), which is slightly above the average.

H1a: Users find the system usable, regardless of their familiarity with graph theory. In our questionnaire we had two questions about graph knowledge. We asked how familiar the subjects are with the following concepts from graph theory: *Vertices and Edges* and *Hyperedges*. Figure 15 shows the distribution of the results.



(a) Distribution for the item *Vertices and Edges* (b) Distribution for the item *Hyperedges*

Figure 15: Question: *How familiar are you with the following concepts from graph theory?* - The knowledge about the basics is bimodal and only few subjects are familiar with hyperedges.

The answer *Vertices and Edges* has a mean of 3.71 ($SD = 1.94$) while following a bimodal distribution. This means that the subjects are either completely familiar or not at all familiar with graph theory. The answer *Hyperedges* has a mean of 2.32 ($SD = 1.46$) which is very low. We expected that usability is independent of familiarity with graph theory and were not able to find a correlation between the SUS-Score and

graph theory basics ($\rho = 0.023, p = 0.878$) nor between the SUS-Score and hyperedge knowledge ($\rho = 0.106, p = 0.464$) in our sample. Therefore, we conclude that the usability of Wust is independent from graph theory familiarity.

H1b: Users find the system usable, regardless of their self-efficacy using technology. The questionnaire contained eight questions to determine the subject's self-efficacy using technology. The questions form a 8-item Likert scale. Only 41 of 51 participants completely answered all these questions. The scale shows a good reliability in our sample ($Cronbach's\alpha = 0.873$) and has a mean of 3.80 ($SD = 0.76$) which means that most participants show a good self-efficacy using technology. As expected, we were not able to find a correlation to the SUS-Score ($r = 0.073, p = 0.650$). This indicates that users find the system usable, regardless of their self-efficacy using technology.

H1c: Users find the system usable, regardless of prior experiences with collaborative online platforms. We asked the subjects about their usage frequency of a set of online platforms: *Internet Forums or Facebook, Mailing Lists, Wikis, Reddit, Stack Overflow, IRC* and *Issue Trackers*. The results form a 7-item Likert scale which show an acceptable reliability ($Cronbach's\alpha = 0.731$). The scale has a mean of 4.02 ($SD = 1.20$), which means that the usage frequency of online platforms is about one time per week in our sample. We interpret the usage frequency as in indicator for prior experience with online platforms. We were not able to find a significant correlation between the SUS-Score and online platform usage frequency ($r = -0.198, p = 0.163$) and can say that users find the system usable, regardless of usage frequency of online platforms and therefore prior experiences with these.

H1d: Users find the system usable, regardless of their sex. The subjects consisted of 15 females and 35 males, which means that the threshold for undetectable differences is $d = 1.14$. The mean of the SUS-Scale for males is 4.55 ($SD = 0.68$) and for females 4.34 ($SD = 0.96$). The T-test for independent variables did not show any difference between the sexes ($T(48) = 0.891$, $p = 0.377$, variance equality: $F = 2.179$, $p = 0.146$). We conclude that there is no difference in perceived usability according to the sexes.

H1e: Users find the system usable, regardless of their age. The subjects of our sample have an average age of 30.90 ($SD = 12.54$). We were not able to find a significant correlation between the SUS-Score and age ($r = 0.242, p = 0.087$). Therefore, we conclude that all users find the system usable, regardless of their age.

The results of the evaluation of the hypotheses **H1** and **H1a-e** show that Wust has a good perceived usability, regardless of sex, age, online platform experience, self-efficacy using technology and familiarity with graph theory. Next we evaluate the hypotheses where the concepts of community moderation and voting mechanisms rely upon.

H2: Users are intrinsically motivated to correct incorrect information. The subjects were asked if they would like to correct different kinds of mistakes when they encounter them. The mean motivation to correct wrong information is 4.58 ($SD = 1.49$). Subjects want to correct wrong connections with a mean of 4.20 ($SD = 1.18$) and punctuation mistakes with a mean of 4.02 ($SD = 1.66$). These values let us conclude that the majority of subjects has an intrinsic motivation to correct incorrect information and mistakes. This result supports the decision for community moderation systems.

H3: Users agree that empowered users are allowed to edit and reconnect posts. Asking the subjects if they agree that empowered users should be able to edit their posts and corresponding connections gave a positive result. The mean for the acceptance for post editing is 4.34 ($SD = 1.26$) and for post reconnection 4.46 ($SD = 0.99$). This result supports the decision for a moderation systems in general.

H4: A collaborative moderation system works by users voting on the changes made by others. When asking subjects who should have the permissions to edit and reconnect the posts of users, they agreed most that these users should be elected by the community ($M = 4.81, SD = 1.28$) and that their past corrections have been rated positively by the community ($M = 4.49, SD = 1.29$). This means that the community wants to decide who edits their posts. It supports our hypothesis to give users permissions based on the rating of their past actions. It is interesting that the subjects also agreed to give editing permissions to the discussion initiator ($M = 4.31, SD = 1.47$), moderators designated by the site owner ($M = 4.20, SD = 1.47$) and users whose posts are rated positively ($M = 4.18, SD = 1.32$). They disagreed with giving editing permissions to every user ($M = 2.63, SD = 1.48$) or users who are members of the community since a long time ($M = 2.38, SD = 1.06$).

H5: Users only vote on positively outstanding posts. The subjects had to answer questions about their voting habits. Subjects agree to only vote on outstanding posts ($M = 4.41, SD = 1.14$) and also prefer voting positively ($M = 4.06, SD = 1.42$) rather than negatively ($M = 2.45, SD = 0.93$). Interestingly the voting activity of users is

quite low. Users do not vote regularly ($M = 3.66, SD = 1.45$) and are indifferent about voting at all. This supports our hypothesis, but the results about voting activity show that this hypothesis is incomplete and does not justify the decision to create a quality estimation solely based on voting.

H6: Users associate upvoting with quality and content discovery. In the questionnaire we asked the subjects what should happen when a post is upvoted. Users agreed that the post should be better discoverable in its context ($M = 4.88, SD = 1$), and that it should be classified as high quality ($M = 4.82, SD = 1.18$). They also agreed that the author should get more moderation permissions ($M = 4.37, SD = 1.20$) and that upvoting is a reading recommendation for others ($M = 4.29, SD = 1.32$). These results validate our hypothesis that users associate voting with quality, but show that discovery is equally important.

5.4. Discussion

With the evaluation data we were able to confirm our hypotheses as expected. The subjects perceived the system as usable (**H1**) and the usability is independent of other important dimensions (**H1a-e**). This reflects our impression from the qualitative feedback. It is important that the subjects needed an explanation to understand the concepts. But after learning the system they were able to use it and found it useful. We were able to create a short tutorial that guides the user through the most important features and explains the most important concepts of Wust. In general the reception was positive. Users pointed out problems with the system and had good ideas which we were able to implement in many cases.

The validated hypotheses justify the use of our voting based community moderation system. Many subjects have an intrinsic motivation to correct different kinds of mistakes and wrong information in the posts of other users (**H2**). They are allowed to do that, because subjects agree, that users whose corrections have been rated as good by the community should be able to edit and reconnect posts (**H3,H4**).

The hypothesis **H5** revealed that users do not vote regularly, which questions the decision to base the post quality on voting. This needs to be investigated further in the future. Hypothesis **H6** shows that users are aware of the relationship between voting, quality and content discovery.

None of our subjects wrote a salutation in a post. This was a design goal of the *Start Discussion* Dialog. On Stack Overflow users often write salutations, but the moderators are encouraged to remove them.

The Net-Promoter-Score was slightly negative. In the 51 subjects, there were 12 promoters and 22 detractors, which results in a Net-Promoter-Score of -19.61%. We conclude that our system leaves room for improvement and future research.

In the second user test one subject told us that:

> *"The system is fun to work with. The graph view is enjoyable and playful. The votestream gives me a feeling of productivity."*

The tutorial on the website led users to create arbitrary content to get familiar with the features and concepts of the system, which in most cases was arbitrary and did not make any sense. Many noted that they would like to try this on real problems and projects. One subject gave us this feedback in the questionnaire:

> *"I think this is a good idea. How well it really works will be revealed after I could use it for questions/topics which are important to me. But after this small demonstration I would definitely like to try it."*

It would be interesting to test the system in real projects and political discussions with many users who are already familiar with the system.

6. Conclusion and Future Work

There is a need for a scaling discussion system. We developed the website-based discussion system called Wust to address the scaling issues of existing systems with argument mapping and a focus on quality on different levels to reduce redundancy and provide overview. The discussion model is based on hypergraphs, where posts can respond to multiple other posts or the relation between two posts. Posts are implicitly classified by the tags on their outgoing relations. For example, a post can be a *pro* and a *contra* argument at the same time.

To address the problem of redundancy and enhance post quality we implemented a community moderation system. Users can suggest changes to posts of others. The community then has to decide if this change should be accepted or rejected. The moderation system was designed to scale in the number of users. It should work for small groups as well as very large groups.

Wust implements a collaborative tagging system with multiple inheritance and synonyms to make the posts better discoverable and remove redundancy in the tag section of posts. To indicate quality, users can vote on posts in respect to their context-tags or posts they are responding to. Whenever posts are displayed as a list, they are sorted by their estimated quality. Post quality is estimated with the goal to not be confused with popularity. This is approximated by counting the unique views of users who did not upvote a post as downvotes.

We evaluated our system with two user tests and one questionnaire. The hypotheses we had were confirmed and justify almost all of our design decisions: Users are able to work with and understand discussions as a graph including hyperrelations. Users also have intrinsic motivation to correct the posts of others, while allowing community selected users to correct their own posts. However, it turned out that users are not voting regularly, which questions the approach of basing the quality estimation only on vote counts.

While developing Wust we were using it as an issue tracking system itself. This helped us to find more UI inconsistencies and showed what was missing for productive usage. After some time it worked very well to track bugs and ideas and relate them to each other.

While identifying and addressing many scaling problems the system has its limitations. The graph view only works until around 40 posts. From that point on the overview gets lost. The next step is to investigate in quality filtering in the graph view as

well as providing radical abstraction mechanisms. This could be achieved with nested graphs where multiple posts are combined to one meta post. These meta posts would behave like regular posts and therefore can be connected and nested again, forming an abstraction tree on top of the graph. The different abstraction layers could provide a better overview and detailed views at the same time.

An approach for nodes with high degree – like many ideas proposed to a very general question – could be solved by providing a binary search on the set of posts over the context-tags. The ideas would then be tagged with preconditions. The search could ask the user if the post one is looking for is tagged with the asked tag or not. By showing the tag with the distribution closest to 50% tagged vs 50% not tagged, the binary search achieves a high efficiency and the post can be found in logarithmic time.

Since users are not voting regularly and only voting on special posts, the quality estimation could be adapted to reflect this. One should investigate other sources of user behavior, such as responding, reconnecting and editing of posts to approximate quality. Users are also biased when selecting quality by the author of the post. It should be examined if the quality which users estimate for a post is influenced by knowing the author of a post.

It would be interesting to know how community moderators can handle redundancy more efficiently. This includes identifying different types of redundancy and providing appropriate tools to handle them.

An idea for automatic cleanup for graph based discussions is to imitate a biological brain. It strengthens the areas of high activity and removes low activity synapses automatically to save energy consumption. The same could be investigated for posts of low user activity.

On the implementation side, we developed a graph database abstraction to be flexible enough in changing the database model. The abstraction exposes a REST API to manipulate the graph. It turned out that it is a lot of effort to request a specific piece of data form the client, since one has to expose the needed data manually via the REST API. It would be better to have some kind of query language similar to GraphQL[15] in the client to save work and reduce sources of errors.

Wust provides a good base for working on these future ideas and developed an open source library to start with. At the same time it serves as a discussion platform for further research on the topic.

[15]https://facebook.github.io/graphql

Overall we received a positive feedback from our subjects and are certain that research and development in this field leads to a truly scaling discussion system helping humanity with solving its complex problems.

References

[AA14] L. de Alfaro and B. Adler. “Content-Driven Reputation for Collaborative Systems”. In: *Trustworthy Global Computing*. Springer, 2014, pp. 3–13.

[And+12] A. Anderson et al. “Discovering Value from Community Activity on Focused Question Answering Sites: A Case Study of Stack Overflow”. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2012, pp. 850–858.

[ARH07] T. Ahn, S. Ryu, and I. Han. “The impact of Web quality and playfulness on user acceptance of online retailing”. In: *Information & Management* 44.3 (2007), pp. 263–275.

[Bet+08] N. Bettenburg et al. “What Makes a Good Bug Report?” In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM. 2008, pp. 308–318.

[BOH11] M. Bostock, V. Ogievetsky, and J. Heer. “D^3 Data-Driven Documents”. In: *Visualization and Computer Graphics, IEEE Transactions on* 17.12 (2011), pp. 2301–2309.

[Bro96] J. Brooke. “SUS-A quick and dirty usability scale”. In: *Usability Evaluation in Industry* 189.194 (1996), pp. 4–7.

[CAP08] K. Chatterjee, L. de Alfaro, and I. Pye. “Robust Content-Driven Reputation”. In: *Proceedings of the 1st ACM Workshop on AISec*. ACM. 2008, pp. 33–42.

[CDL14] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec. “How Community Feedback Shapes User Behavior”. In: *ICWSM* (2014).

[Cos+05] D. Cosley et al. “How Oversight improves Member-Maintained Communities”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2005, pp. 11–20.

[Cos+06] D. Cosley et al. “Using Intelligent Task Routing and Contribution Review to Help Communities Build Artifacts of Lasting Value”. In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM. 2006, pp. 1037–1046.

[DN11] N. Diakopoulos and M. Naaman. “Towards Quality Discourse in Online News Comments”. In: *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work*. ACM. 2011, pp. 133–142.

[Gil13] E. Gilbert. “Widespread Underprovision on Reddit”. In: *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*. ACM. 2013, pp. 803–808.

[HBC15] F. Hendrikx, K. Bubendorfer, and R. Chard. “Reputation systems: A survey and taxonomy”. In: *Journal of Parallel and Distributed Computing* 75 (2015), pp. 184–197.

[HG06] P. Heymann and H. Garcia-Molina. “Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging Systems”. In: *InfoLab Technical Report* (2006).

[Hil09] M. Hilbert. “The Maturing Concept of E-Democracy: From E-Voting and Online Consultations to Democratic Value Out of Jumbled Online Chatter”. In: *Journal of Information Technology & Politics* 6.2 (2009), pp. 87–110.

[III14] F. A. III. *Reddit’s ’xkcd’ community is fighting to drive out its top moderator*. [Accessed 17.11.2015]. Jan. 2014. URL: `http://www.dailydot.com/news/xkcd-reddit-moderator-drama/`.

[Kar15] J. Karoff. “Modeling Open Discourse in a Structured Collaborative Discussion System”. MA thesis. RWTH Aachen University, 2015.

[Kle11] M. Klein. “How to Harvest Collective Wisdom on Complex Problems: An Introduction to the MIT Deliberatorium”. In: *Center for Collective Intelligence Working Paper* (2011).

[Las14] C. Last. “Global Brain and the Future of Human Society”. In: *World Future Review* (2014), p. 1946756714533207.

[LPC12] R. Lotufo, L. Passos, and K. Czarnecki. “Towards Improving Bug Tracking Systems with Game Mechanisms”. In: *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*. IEEE Press. 2012, pp. 2–11.

[LR04] C. Lampe and P. Resnick. “Slash(dot) and Burn: Distributed Moderation in a Large Online Conversation Space”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2004, pp. 543–550.

[Lud+04] P. J. Ludford et al. “Think Different: Increasing Online Community Participation Using Uniqueness and Group Dissimilarity”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2004, pp. 631–638.

[Luu] D. Luu. *Why HN Should Use Randomized Algorithms*. [Accessed 17.11.2015]. URL: `http://danluu.com/randomize-hn/`.

[MAT13] L. Muchnik, S. Aral, and S. J. Taylor. “Social Influence Bias: A Randomized Experiment”. In: *Science* 341.6146 (2013), pp. 647–651.

[Mil13] R. Mills. “Distributed Moderation Systems: An Exploration of Their Utility and the Social Implications of Their Widespread Adoption”. PhD thesis. Lancaster University, 2013.

[MM06] G. Macgregor and E. McCulloch. “Collaborative Tagging as a Knowledge Organisation and Resource Discovery Tool”. In: *Library Review* 55.5 (2006), pp. 291–300.

[Mun09] R. Munroe. “reddit’s new comment sorting system”. In: *The Reddit Blog. Reddit, Inc* (2009).

[PL15] M. Paskuda and M. Lewkowicz. “Anonymous Quorans are still Quorans, just anonymous”. In: *Proceedings of the 7th International Conference on Communities and Technologies*. ACM. 2015, pp. 9–18.

[Pon+14] L. Ponzanelli et al. “Understanding and Classifying the Quality of Technical Forum Questions”. In: *2014 14th International Conference on Quality Software (QSIC)*. IEEE. 2014, pp. 343–352.

[Reda] Reddit. *How do I become a moderator?* [Accessed 17.11.2015]. URL: `https://reddit.zendesk.com/hc/en-us/articles/205192505-How-do-I-become-a-moderator-`.

[Redb] Reddit. *How is a post’s or comment’s score determined?* [Accessed 17.11.2015]. URL: `https://reddit.zendesk.com/hc/en-us/articles/204511559-How-is-a-post-s-or-comment-s-score-determined-`.

[Redc] Reddit. *What is karma?* [Accessed 17.11.2015]. URL: `https://reddit.zendesk.com/hc/en-us/articles/204511829-What-is-karma-`.

[Rei03] F. F. Reichheld. “The One Number You Need to Grow”. In: *Harvard Business Review* 81.12 (2003), pp. 46–55.

[Ric14] A. Richterich. “‘Karma, precious karma!’ Karmawhoring on Reddit and the Front Page’s Econometrisation”. In: *Journal of Peer Production* 4 (2014), p. 29.

[Sen+07] S. Sen et al. “The Quest for Quality Tags”. In: *Proceedings of the 2007 international ACM Conference on Supporting Group Work*. ACM. 2007, pp. 361–370.

[SRC08] P. Schugerl, J. Rilling, and P. Charland. “Mining Bug Repositories–A Quality Assessment”. In: *Computational Intelligence for Modelling Control & Automation, 2008 International Conference on*. IEEE. 2008, pp. 1105–1110.

[Sto15] G. Stoddard. “Popularity and Quality in Social News Aggregators: A Study of Reddit and Hacker News”. In: *Proceedings of the 24th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee. 2015, pp. 815–818.

[Sun11] J. Sun. “Why are Bug Reports Invalid?” In: *Verification and Validation (ICST), 2011 IEEE Fourth International Conference on Software Testing*. IEEE. 2011, pp. 407–410.

[SVG08] F. M. Suchanek, M. Vojnovic, and D. Gunawardena. “Social Tags: Meaning and Suggestions”. In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. ACM. 2008, pp. 223–232.

[Vas+14] B. Vasilescu et al. “How Social Q&A Sites are Changing Knowledge Sharing in Open Source Software Communities”. In: *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM. 2014, pp. 342–354.

[WH08] F. Wu and B. A. Huberman. “How Public Opinion Forms”. In: *Internet and Network Economics*. Springer, 2008, pp. 334–341.

[WS96] R. Y. Wang and D. M. Strong. “Beyond Accuracy: What Data Quality Means to Data Consumers”. In: *Journal of Management Information Systems* (1996), pp. 5–33.

[YCo] YCombinator. *Hacker News FAQ*. [Accessed 17.11.2015]. URL: `https://news.ycombinator.com/newsfaq.html`.

[Zha+11] D. Zhang et al. “How to Count Thumb-Ups and Thumb-Downs: User-Rating based Ranking of Items from an Axiomatic Perspective”. In: *Advances in Information Retrieval Theory*. Springer, 2011, pp. 238–249.

A. First User Test: Tasks

Aufgaben

1. Du siehst einen Beitrag zu einer Diskussion. Lies ihn.

2. Beschreibe, was du sonst noch siehst.

3. Navigiere zur Detailansicht einer Antwort.

4. Du findest eine Antwort unpassend. Entferne sie aus der Diskussion.

5. Stelle eine Frage zum Beitrag "Was denkst du über Wäsche?": „Normale oder Unterwäsche?“.

6. Du möchtest mehr Übersicht bekommen, wechsele in eine übersichtlichere Ansicht.

7. Beschreibe was du hier siehst.

8. Füge die Idee "Wäsche waschen" zu dem Problem "Meine Wäsche ist schmutzig" hinzu.

9. Auf den Beitrag "Meine Wäsche stinkt" passt deine Antwort auch. Verbinde Problem und Idee.

10. Du hast dich vertan. Entferne die eben erstellte Verbindung.

11. Du hast das Problem "Ich weiß nicht, wie ich eine neue Diskussion starte.". Starte eine neue Diskussion damit.

12. Du möchtest die eben erstellte Diskussion um eine Beschreibung ergänzen. Editiere den Beitrag.

B. Second User Test: Tasks

Usertest

Dies ist eine Benutzerstudie über ein Graphbasiertes Diskussionssystem. Die Studie soll herausfinden, ob die Benutzeroberfläche einfach zu bedienen ist. Während des Usertests wird Bildschirminhalt und Ton als Video aufgezeichnet. Alle Aufzeichnung werden ausschließlich anonymisiert zur Auswertung verwendet und nicht veröffentlicht.

Bitte erledige die Aufgaben der Reihe nach. Wenn du eine Aufgabe nicht erledigen kannst, ist das nicht dein Fehler, sondern ein Fehler der Benutzeroberfläche, die nicht verständlich genug gestaltet wurde. Genau solche Fehler gilt es aufzudecken.

Bitte sprich deine Gedanken laut aus, so dass noch mehr Missverständnisse erkannt werden können. Zum Beispiel: **"Wenn ich hier klicke, müsste sich die Suche öffnen"** oder **"Ich bin mir gerade nicht sicher, was ich anklicken soll"**.

Aufgaben

Strukturierung einer Diskussion

1. Vor dir siehst du den Beitrag **"Planung des Betriebsausflugs der Firma Wust"**. Beschreibe, wie der Beitrag dargestellt wird. Welche Informationen werden noch dargestellt? Beschreibe auch, was passieren würde, wenn du auf die Buttons und Links klicken würdest.
2. Logge dich als Benutzer **"John"** (großes J) mit dem Passwort **"geheim"** (alles klein) ein.
3. Navigiere zum Task **"Veranstaltungsort finden"**. Navigiere weiter zum Ziel **"Unterbringung von ca. 50 Personen"**. Gehe zurück zum Start der Diskussion.
4. Stelle eine Frage zum Beitrag **"Planung des Betriebsausflugs der Firma Wust"**: Du würdest gerne wissen **"Was wurde letztes Jahr gemacht?"**.
5. Du möchtest mehr Übersicht bekommen. Wechsle in die Graph-Ansicht. Beschreibe, ob diese Ansicht wirklich übersichtlicher ist.
6. Füge die Idee **"Skifahren in den Alpen"** zu dem Task **"Veranstaltungsort finden"** hinzu.
7. Auf den Beitrag **"Zeitrahmen vereinbaren"** passt deine Idee auch. Verbinde Idee und Task.
8. Du hast dich vertan. Entferne die eben erstellte Verbindung.
9. Du würdest gerne Reis kochen und fragst dich **"Wie viel Reis brauche ich pro Person?"**. Starte eine neue Diskussion im Kontext "Haushalt".
10. Du möchtest die eben erstellte Diskussion um eine Beschreibung (z.B. Langkornreis) ergänzen. Editiere den Beitrag.

Fortgeschrittene Strukturierung einer Diskussion

1. Suche eine Diskussion zum Thema **"Urlaub"** und sieh sie dir in der Graph-Ansicht an.
2. Füge das Argument **"Mit dem Fahrrad kann man nicht über Wasser fahren"** hinzu.
3. Füge das Argument **"Fahrradfahren ist umweltfreundlich"** hinzu.

Moderation

1. In der Frage **"Mit welchem Verkehrsmittttel fahre ich am besten zur Arbeit"** ist ein Rechtschreibfehler. Korrigiere ihn. Warum ist die Änderung nicht sofort sichtbar? Wenn dir dies nicht klar ist, öffne nochmal das "Bearbeiten"-Fenster und wirf einen Blick auf die Grüne Leiste unten.
2. Klicke in der Navigationsleiste auf **"Moderation"**. Hier siehst du Änderungen von anderen Benutzern. Du kannst entscheiden, ob diese akzeptiert oder abgelehnt werden. Bewerte zwei Änderungen.
3. Suche nach einer Frage mit dem Stichwort **"reparieren"** und öffne sie.
4. Du findest eine Antwort unpassend. Entferne sie aus der Diskussion. Navigiere zurück zur Frage.
5. Du siehst, dass nach der Art der Schaltung gefragt wurde. Du kennst **"Barbera"** persönlich und weißt natürlich, dass sie ein Fahrrad mit einer Nabenschaltung besitzt. Ergänze die Frage von Barbera. Was bedeutet der Grüne Balken unter dem Beschreibungs-Formular?
6. Die Frage nach der Art der Schaltung ist jetzt beantwortet und somit überflüssig. Was ist zu tun?

C. Questionnaire

Wust - Ein Graphbasiertes Diskussionssystem

Vielen Dank, dass Sie sich entschieden haben an der Benutzerstudie zu "Wust" teilzunehmen.
Die Teilnahme an dieser Studie ist freiwillig und dient der Verbesserung des Software-Systems "Wust". Die erhobenen Daten werden vertraulich behandelt und nur für wissenschaftliche Zwecke verwendet. Sämtliche Informationen werden anonymisiert gespeichert und können auf Ihren Wunsch hin jeder Zeit gelöscht werden.
Es besteht keine Möglichkeit auf Ihre Person Rückschlüsse zu ziehen.

Dieser Fragebogen ist Teil einer Benutzerstudie über Wust - einem graphbasierten Diskussionssystem. Er setzt voraus, dass Sie das System bereits ausprobiert haben. Sollte dies nicht der Fall sein, können Sie das unter folgendem Link nachholen:

Wust ausprobieren

Dies ist ein Test-System, sie können nach Belieben alles ausprobieren.

Nachdem Sie sich mit dem System vertraut gemacht haben, können Sie mit dem Fragebogen fortfahren.

Verantwortlich für die Durchführung dieser Studie:
Prof. Dr. Martina Ziefle
Dr. André Calero Valdez
Lehrstuhl für Kommunikationswissenschaft
RWTH Aachen University

Prof. Dr. Ulrik Schroeder
Christoph Greven, M.Sc. RWTH
Lehr- und Forschungsgebiet Informatik 9
RWTH Aachen University

Wust - Ein Graphbasiertes Diskussionssystem

Zunächst würden wir gerne etwas über Ihre Person erfahren wollen.

* 1. Wie alt sind Sie?

* 2. Geschlecht

Männlich Weiblich Keine Angabe

3. Wie häufig nutzen Sie folgende Internet-Plattformen?

	Täglich	2-3x pro Woche	1x pro Woche	1x pro Monat	2-3x pro Jahr	Seltener	Nie
Foren (Bulletin boards) oder Facebook							
Mailing Listen / Verteiler							
Wikis (z.B. Wikipedia)							
Reddit							
StackOverflow							
IRC							
Issue/Bug Tracker							

4. Wenn Ich eben genannte Plattformen nutze, bin ich ...
(Wenn Sie nicht angemeldet sind, können Sie die Letzten 3 Fragen auslassen)

	Stimme vollständig zu	Stimme zu	Stimme eher zu	Lehne eher ab	Lehne ab	Lehne vollständig ab
passiv, ich lese nur						
zurückhaltend und antworte selten						
sehr aktiv und starte neue Diskussionen						
sehr hilfsbereit und beantworte offene Fragen						

5. Wie vertraut sind Sie mit folgenden Konzepten der Graphentheorie?

	Sehr vertraut	Vertraut	Eher Vertraut	Eher nicht vertraut	nicht vertraut	Gar nicht vertraut
Knoten und Kanten						
Hyperkanten						

6. Sie haben gerade eine Diskussionsplattform genutzt. Wie sehr treffen folgende Aussagen auf das System, oder den Bereich des Systems den Sie kennen, zu?

	Stimme vollständig zu	Stimme zu	Stimme eher zu	Lehne eher ab	Lehne ab	Lehne vollständig ab
Ich kann mir sehr gut vorstellen, das Sytem regelmäßig zu nutzen.	○	○	○	○	○	○
Ich empfinde das System als unötig komplex.	○	○	○	○	○	○
Ich finde das System als einfach zu nutzen.	○	○	○	○	○	○
Ich denke, dass ich technischen Support brauchen würde, um das System zu nutzen.	○	○	○	○	○	○
Ich finde, dass die verschiedenen Funktionen des Systems gut integriert sind.	○	○	○	○	○	○
Ich finde, dass es im System zu viele Inkonsistenzen gibt.	○	○	○	○	○	○
Ich kann mir vorstellen, dass die meisten Leute das System schnell zu beherrschen lernen.	○	○	○	○	○	○
Ich empfinde die Bedienung als sehr umständlich.	○	○	○	○	○	○
Ich habe mich bei der Nutzung des Systems sehr sicher gefühlt.	○	○	○	○	○	○
Ich musste eine Menge Dinge lernen, bevor ich mit dem System arbeiten konnte.	○	○	○	○	○	○

7. Diskussion

	Stimme vollständig zu	Stimme zu	Stimme eher zu	Lehne eher ab	Lehne ab	Lehne vollständig ab
Die Anordnung der Beiträge hilft mir, den Zusammenhang zwischen Beiträgen zu erkennen.	○	○	○	○	○	○
Die Anordnung der Beiträge hilft mir, der Diskussion zu folgen.	○	○	○	○	○	○
Die Anordnung der Beiträge hilft mir, alle wichtigen Argumente zu erfassen.	○	○	○	○	○	○
Ich kann meine Gedanken in dem System verfassen.	○	○	○	○	○	○
Es fällt mir leicht, meinen Beitrag in die Diskussion einzuordnen.	○	○	○	○	○	○
Es ist mir wichtig, dass Beiträge sinnvoll miteinander verbunden sind.	○	○	○	○	○	○
Das System macht es mir leicht, mich an der Diskussion zu beteiligen.	○	○	○	○	○	○
Ich finde es angenehm, dass ich Beiträge aus einer Diskussion mit den Beiträgen aus einer anderen Diskussion verbinden kann.	○	○	○	○	○	○

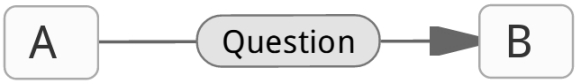
8. Es ist mir wichtig, dass Beiträge nach folgenden Kritierien angeordnet werden:

	Stimme vollständig zu	Stimme zu	Stimme eher zu	Lehne eher ab	Lehne ab	Lehne vollständig ab
Strikt chronologisch	○	○	○	○	○	○
Kontext, z.B. in der Nähe des beantworteten Beitrags	○	○	○	○	○	○
Qualitätsbewertung anderer Benutzer	○	○	○	○	○	○

Pfeil- und Leserichtung

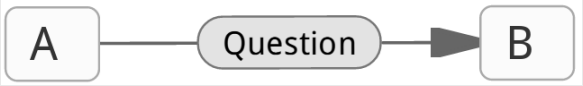
Im folgenden Abschnitt wollen wir heraus finden, wie verschiedene Aspekte in Wust verstanden werden.

9. Wie stehen die Beiträge A und B in Verbindung?

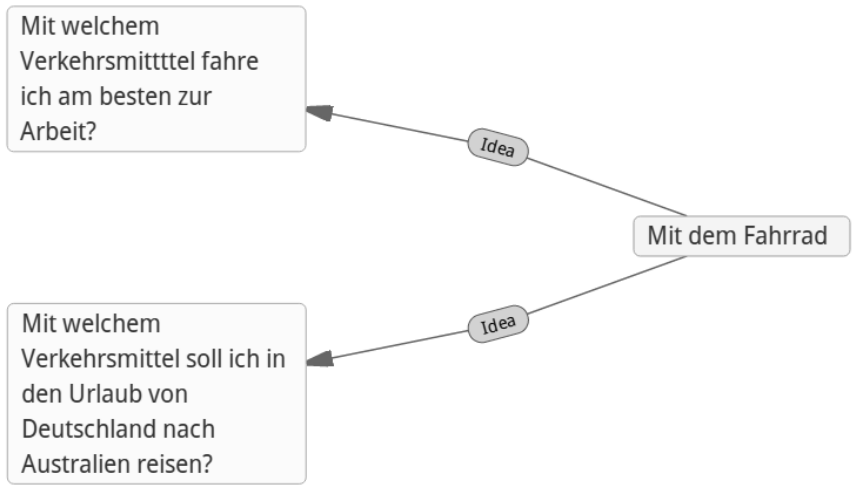


- ◯ A ist eine Frage zu B
- ◯ B ist eine Frage zu A

10. A ist eine Frage zu B. Welche Darstellung empfinden Sie als richtig?

	Stimme vollständig zu	Stimme zu	Stimme eher zu	Lehne eher ab	Lehne ab	Lehne vollständig ab
	◯	◯	◯	◯	◯	◯
	◯	◯	◯	◯	◯	◯
	◯	◯	◯	◯	◯	◯
	◯	◯	◯	◯	◯	◯

11. An welchen Punkt würden Sie in folgendem Beispiel das Argument "Mit dem Fahrrad kann man nicht über Wasser fahren" anhängen?



(Mehrfachantworten sind erlaubt)

- [ ] An die **Frage** "Mit welchem Verkehrsmittttel fahre ich am besten zur Arbeit?"
- [ ] An die **Frage** "Mit welchem Verkehrsmittel soll ich in den Urlaub von Deutschland nach Australien reisen?"
- [ ] An die **Idee** "Mit dem Fahrrad"
- [ ] An die **Verbindung** zwischen "Mit welchem Verkehrsmittttel fahre ich am besten zur Arbeit?" und "Mit dem Fahrrad"
- [ ] An die **Verbindung** zwischen "Mit welchem Verkehrsmittel soll ich in den Urlaub von Deutschland nach Australien reisen?" und "Mit dem Fahrrad"

Bewertungen

12. Sie bewerten einen Beitrag positiv. Was sollte passieren?

	Stimme vollständig zu	Stimme zu	Stimme eher zu	Lehne eher ab	Lehne ab	Lehne vollständig ab
Der Beitrag wird als qualitativ hochwertig eingestuft.	○	○	○	○	○	○
Der Autor des Beitrags wird anerkannter und bekommt mehr Moderationsrechte.	○	○	○	○	○	○
Der Beitrag wird anderen von mir zum Lesen empfohlen.	○	○	○	○	○	○
Ich abonniere alle weiteren Beiträge der Diskussion.	○	○	○	○	○	○
Der Beitrag ist in seinem Kontext besser auffindbar.	○	○	○	○	○	○

13. Ich bewerte...

	Stimme vollständig zu	Stimme zu	Stimme eher zu	Lehne eher ab	Lehne ab	Lehne vollständig ab
selbstverständlich/direkt beim Lesen	○	○	○	○	○	○
ausschließlich besondere Beiträge	○	○	○	○	○	○
Beiträge lieber positiv	○	○	○	○	○	○
Beiträge lieber negativ	○	○	○	○	○	○
am liebsten gar nicht	○	○	○	○	○	○

Moderation

14. Welche Nutzer sollten Rechte für die Bearbeitung von Beiträgen und Verbindungen erhalten?

	Stimme vollständig zu	Stimme zu	Stimme eher zu	Lehne eher ab	Lehne ab	Lehne vollständig ab
Nutzer, die lange dabei sind	○	○	○	○	○	○
Nutzer, die vom Seitenbesitzer als Moderator bestimmt wurden	○	○	○	○	○	○
Nutzer, die von der Community als Moderator bestimmt wurden	○	○	○	○	○	○
Nutzer, deren *Beiträge* von der Community positiv bewertet wurden	○	○	○	○	○	○
Nutzer, deren *Korrekturen* von der Community positiv bewertet wurden	○	○	○	○	○	○
Nutzer, die die jeweilige Diskussion gestartet haben	○	○	○	○	○	○
Alle Nutzer	○	○	○	○	○	○

15. Korrekturen

	Stimme vollständig zu	Stimme zu	Stimme eher zu	Lehne eher ab	Lehne ab	Lehne vollständig ab
Wenn ich falsche Verbindungen zwischen Beiträgen sehe, würde ich die Korrekturen selbst vornehmen.	○	○	○	○	○	○
Wenn ich Formfehler (z.B. Rechtschreib-, Kommafehler) in Beiträgen sehe, würde ich sie gerne korrigieren.	○	○	○	○	○	○
Wenn ich inhaltliche Fehler (z.B. falsche Information) in Beiträgen sehe, würde ich sie gerne korrigieren.	○	○	○	○	○	○
Ich finde es sinnvoll, wenn ausgewählte Nutzer (vgl. Frage 14) die Beiträge anderer neu verbinden können.	○	○	○	○	○	○
Ich finde es sinnvoll, wenn ausgewählte Nutzer (vgl. Frage 14) die Beiträge anderer bearbeiten können.	○	○	○	○	○	○

16. Bitte nehmen sie Stellung zu den folgenden Aussagen:

	stimme vollständig zu	stimme zu	stimme eher zu	lehne eher ab	lehne ab	lehne vollständig ab
Ich kann ziemlich viele der technischen Probleme, mit denen ich konfrontiert werde, allein lösen.	○	○	○	○	○	○
Technische Geräte sind oft undurchschaubar und schwer zu beherrschen.	○	○	○	○	○	○
Es macht mir richtig Spaß, ein technisches Problem zu knacken.	○	○	○	○	○	○
Weil ich mit bisherigen technischen Problemen gut zurecht gekommen bin, blicke ich auch künftigen technischen Problemen optimistisch entgegen.	○	○	○	○	○	○
Ich fühle mich technischen Geräten gegenüber so hilflos, dass ich die Finger von ihnen lasse.	○	○	○	○	○	○
Auch wenn Widerstände auftreten, bearbeite ich ein technisches Problem weiter.	○	○	○	○	○	○
Wenn ich ein technisches Problem löse, so geschieht es meistens durch Glück.	○	○	○	○	○	○
Die meisten technischen Probleme sind so kompliziert, dass es wenig Sinn hat, sich mit ihnen auseinander zu setzen.	○	○	○	○	○	○

Feedback

17. Wie wahrscheinlich ist es, dass Sie dieses Diskussionssystem einem Freund oder Kollegen weiterempfehlen werden?

Gar nicht wahrscheinlich — Äußerst wahrscheinlich

0	1	2	3	4	5	6	7	8	9	10

18. Wenn Sie noch Anmerkungen zur Befragung haben, können Sie die hier gerne angeben.

19. Über Feedback zu unserem System würden wir uns freuen.