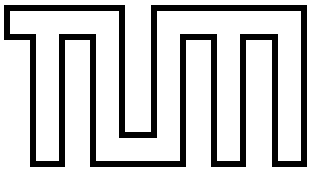


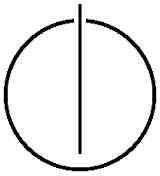
FAKULTÄT FÜR INFORMATIK

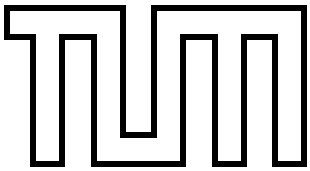
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Evaluating the Usability of a Tag-based, Multi-faceted Knowledge Organization System

Joan Boixadós Sanuy





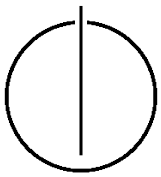
FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Evaluating the Usability of a Tag-based, Multi-faceted Knowledge Organization System

Author:	Joan Boixadós Sanuy
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	M. Sc. Alexander Steinhoff
Date:	August 31, 2012



Ich versichere, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 30. August 2012 Joan Boixadós Sanuy

Acknowledgments

I would like to thank my family for their support as well as everyone who has helped me with the development of this work, especially the participants of the interviews who devoted some of their time to me out of kindness.

M'agradaria agraïr a la meva família el suport rebut així com a tothom que m'ha ajudat en la consecució d'aquest treball, especialment als participants de les entrevistes qui desinteressadament m'han dedicat part del seu temps.

Abstract

With the advent of the Web 2.0, where social communities share large amounts of resources such as bookmarks (Delicious.com) or photographs (Flickr.com), the need for a flexible yet structured knowledge organization system becomes obvious. In this manner, social tagging systems, systems that enable users to collaboratively manage collections of information resources via the assignment of text labels to the resources, have become popular in recent years. Nonetheless, navigation among tags and resources has been found to be limited in these systems. Therefore, a new wave of social tagging systems has appeared to try to find solutions to this limitation. However, the design of an interface for innovative tagging systems that incorporate many new concepts unknown to the users is not trivial. In this work, we conduct an exhaustive evaluation of the interface of one of these new approaches, the multi-faceted context-dependant knowledge organization system TACKO. For such purpose, we first provide the necessary background knowledge concerning social tagging systems, the usability evaluation methods we used and the TACKO tagging system, object of our evaluation. Subsequently, we proceed to present the plan and execution of our evaluation of TACKO's interface with the help of 16 testers, to finally present an exhaustive analysis of the interface, concluding with a list of the main found usability issues as well as possible improvements to solve them.

Contents

1. Introduction

In the last years, tagging systems have become very popular as a means to categorize large amounts of information to ease their future search and retrieval. These systems consist of attaching text labels, the so called *tags*, to resources of a wide and diverse nature, such as bookmarks, photographs, cites, books, videos, and about everything that can be digitally stored. These systems provide a great flexibility in relation to more traditional knowledge organization systems, which has led to their rapid growth in popularity among the Web 2.0. In this scope, they provide the possibility to collaboratively create categorization structures whose result is commonly known as a *folksonomy*. These folksonomies are continuously evolving and do not require to be previously designed, which represents a great advantage. Additionally, users can individually adapt the categorization to suit their own needs.

However, these systems appear to have serious limitations at the time to offer proper navigation options due to the chaotic nature of tags. The lack of human-generated relations among tags and the inconsitent usage of these impede the possibility to offer efficient and structured navigation options to facilitate the browsing through collections of resources. Several different approaches in this direction exist, such as *tagclouds* and *tagclusters*, but they have proved not to be reliable enough to ensure efficient finding and retrieval of resources in large collections because they mainly rely on tag co-occurrences and do not offer users the possibility to improve them with their knowledge.

It is at this point where new knowledge organization systems such as TACKO are proposed. TACKO is a new organization scheme founded in the flexibility of tags but enabling users to represent their knowledge in the form of relations inspired in the concepts of hierarchies and faceted classifications. Thanks to the simple nature of tags it enables us to treat different categorization dimensions independently while still defining hierarchical correspondences among them. Nonetheless, it is not trivial to design an interface that enables users to manage their collections of resources benefiting from all the advantageous properties of the aforementioned schemes this system incorporates.

The goal of this work is to evaluate the usability of the current interface prototype as well as to explore as to which extent do the features help users to understand the main concepts of the underlying system. Primarily, we evaluate how effectively do users use the system to search, organize and discover resources within the system. Additionally, we intend to find out which is the minimum amount of information users need to understand the new concepts introduced by the system and how can this information be transmitted and represented in the interface.

To achieve our goal we develop a series of tests to a total of 16 participants with background in computer science using diverse usability methods such as usability testing, in-

terviews, field observation, thinking aloud and constructive interaction. We define a series of tasks they have to develop to evaluate the practicability of every of the uses of the system: search, organize and discover.

These tests are conducted in four prototypal versions, simplification of the original prototype, that we have especially developed for this work and that enable users to import collections of resources and tags from the social bookmarking site Delicious.com and the image sharing site Flickr.com.

In this document, we first provide the necessary background knowledge regarding the *Tagging Systems* in Chapter 2, with an especial focus on *Social Tagging*, and a very complete general overview of *Usability* and the most commonly applied methods for *Usability Evaluation*, of both types: *Usability Inspection Methods* and *Usability Test Methods*, in Chapter 3. Subsequently, in Chapter 4 we present the *TACKO* tagging system whose interface is object of our evaluation. A very complete explanation of its interface is presented to ease the understanding of the concepts discussed during the analysis in subsequent chapters. In Chapter 5, we talk about the *Plan and execution of the tests*, where we explain the reason for choosing the forementioned usability evaluation methods as well as how have the tests been designed. In Chapter 6, a thorough *Analysis* of the interface is presented together with observations and findings obtained from the tests. Additionally, several possible improvements are proposed as a consequence of the previously mentioned findings. Finally, in *Summary & Outlook* a conclusion of this work is explained together with some general considerations that should be born in mind for future development of the interface.

Part I.

Background Knowledge

2. Tagging Systems

Humans have always been organizing machines and some of the most relevant breakthroughs in human history appeared in order to support us during the development of this task. For example, writing appeared out of the necessity to provide reliable means for transmitting and organizing information when the complexity of trade and administration outgrew the power of memory. We have always been grouping, sorting, tracking and organizing just about everything we encounter. What has changed is the way in how we do it. Since writing first appeared, numerous developments to order and classify information have followed, being a remarkable example Dewey's Decimal Classification System from late 19th Century, still used in libraries today to ease the storage and retrieval of books [Mar06].

Nowadays we think nothing of finding what we need in between tens of billions of webpages indexed by search engines. However, the information landscape has incredibly changed in the last years, and keeps changing every day, where millions of items are generated on an hourly basis not even the most sophisticated search engines can successfully organize and find. It is at this point where a new wave of organizing approaches appear under the name of the *semantic web*. All these approaches rely on the idea that the solution is to let the people do the categorization [Mar06]. Of these, one of the most widespread in recent years are the *Tagging Systems*.

In this chapter, we will present the concept of *Tagging* and how tags can be related using varied *Classification Systems*. In *Social Tagging* we will talk about the fundamental changes tagging heralds in how we manage information today, and the influence that increasingly social information and classification systems is having on tagging [Smi08]. For such purpose, we will briefly present *Delicious* and *Flickr*, two of the main representative examples of social tagging, whose datasets we will be using for the analysis of TACKO's interface proposed in Chapters 5 and 6. In *Tag Presentation Techniques* we will talk about some of the most common practices at the time to present tags within the interfaces and the impact those have on the navigability through them.

2.1. Tagging

Tagging describes the action of attaching one or more keywords or *tags* to a certain resource with the goal to organize content for future navigation and retrieval. This methodology has been used for several years but it has not been until the advent of the Web 2.0 that it became popular. It is in this scope where the strengths of tags have proven to be crucial. Its ease of use and rapid adaptation has let a broader group of users, or should we call them *taggers*, put it in practice with their own collections of resources. Tags' flexibility and universal applicability —they can be used in almost any situation and for any information resource— has also contributed to its popularity on the web.

2.1.1. Classification Systems

Classification systems are a set of different approaches to create relationships between terms, in order to facilitate the categorization of knowledge. In this section, we present some of the most relevant of these approaches: controlled vocabularies, taxonomies, facets and folksonomies.

2.1.1.1. Controlled Vocabularies

Controlled vocabularies are a set of classification systems that aim to organize the information for subsequent retrieval. Controlled vocabularies consist of a restricted list of words or terms used for indexing or categorizing. They are called *controlled* because only terms from such a list might be used for the subject area covered by the specific vocabulary. Also, in cases where used by more than one person, there is still a control over who manages the terms of the list and how these are added to or removed from the set. These vocabularies might grow but always under certain policies. The majority of controlled vocabularies include the *See*-type feature, a cross-reference pointing from non-preferred terms to preferred terms within the list. The main goal of these systems is to ensure the consistency of the classifications, tags, or categorizations, and to guide the user without confusion to where the information they seek is [Hed08]. One of the main advantages of controlled vocabularies is that many of the inherent semantic problems of language can be minimized.

Two of the most common types of controlled vocabularies are [Smi08]:

- **Synonym Rings**: A *synonym ring* relates two or more words with equivalent meanings. These vocabularies are generally used to smooth differences between acronyms, their variations and their fully expanded names. It also reduces variations such as spaces, e.g. *stylesheets* and *style sheets* are matched, hyphens or upper vs lower-case differences. In this approach, all the related terms share the same level of importance [Hed08].
- **Authority Files**: *Authority files* are similar to synonym rings but having a term identified as preferred, which prevails over the other terms in the association. Normally,

the authoritative term is the one showed to the users, and alternative terms cross-reference to the prevailing one. This approach is particularly good to map popular terms, nicknames, variations and abbreviations to the official term. For example, in a music related system we would like to relate *System*, *SOAD* or *systemofadown* to *System Of A Down*.

Recently, the apparition of new sorts of tagging systems such as social cataloging sites like LibraryThing, where users and experts tag books simultaneously, has presented, for the first time, an opportunity to compare expert created library data to tags from an uncontrolled vocabulary created by users [HGM09].

2.1.1.2. Taxonomies

Taxonomy literally means the science of classifying things. Lately, the term *taxonomy* becoame popular to define any hierarchical classification or categorization system. In other words, taxonomies are kind of controlled vocabularies that define hierarchies among their terms [Hed08]. The hierarchies defined in taxonomies are exclusive.

Typical examples of taxonomies are the Linnean system of classifying living species or the Dewey Decimal classification for libraries. A computer file system is also a taxonomy. Independently of if the resources are species, books or files, they all fit in an unambiguous category [GH05] making it easier to both, categorization and retrieval. However, hierarchies inherently involve limitations, such as the difficulty to assign resources to single nodes in the trees. Limitation that arises as well at the time to search for the resource. Moreover, hierarchies oblige to access to the maximum level of specificity to find the resources. In spite of these limitations, hierarchies avoid confusions as to the relevance of resources within each scope and they are easily understood by the audience because they mirror the real world [MNS12a].

Building taxonomies for social tagging systems may be useful for the classification and organization of resources, but taxonomies are generally rigid, conservative and centralized [Qui05]. As we said earlier, items do not always fit exactly inside one category and this decision is always taken from the cataloguer's point of view, which might differ from other users of the system. Thus, totally ignoring the variety of user needs and views [GLYH]. It is at this point where folksonomies appear as a solution for the new aspects social tagging involves (See Section 2.1.1.4). Despite the success of these new approach, there have been some studies proposing the creation of collaborative communal hierarchical taxonomies that behave with a high level of similarity to folksonomies [HGM06].

2.1.1.3. Facets

Faceted classification is a classification system that organizes resources by their relevant properties. Facets represent different dimensions in which resources can be classified such as time or location. The concept of facets was introduced by Ranganathan in the early 1930s though it was not until the advent of the Web 2.0 that they draw considerable attention [Smi08].

Facets are defined by sets of concepts that are attached to different resources and can define hierarchies to one another within the facet [MNS12a]. In general, individual resources can be related to more than one of the concepts of the facet, though this does not need to necessarily be the rule. This fact is also constrained by the design of the particular application. For example, *Oktoberfest* would be attached to both *september* and *october* within the facet representing the group of months.

Nowadays, some tagging systems implement ways to manage these facets within the system, with the goal to improve the navigability through tags and to boost the semantic value of the system's tags. Some of the benefits facets incorporate to tagging systems imply an improvement of the precision of tags by avoiding possible confusions, a string of text is way more meaningful within a context, and facets provide this context that individual tags do not have. For example, *man* has a way different meaning if among *woman*, *girl* and *boy*, than if among *volvo*, *mercedes* and *scania*, where it makes reference to the truck brand. Moreover, facets improve the findability of tags as they ease the browsing through tags that are grouped because of their similar concepts. In any case, it must be born in mind that the categorizing work made by few users will have a great impact in other's user experience of the system [Smi08].

2.1.1.4. Folksonomies

The term folksonomy appears as a conjunction of *folk* (people) + *taxonomy* (classification management) and basically means *user-generated classification*. The term was coined by Thomas Vander Wal to baptize the widespread practice of collaborative categorization using freely chosen keywords by groups of people cooperating spontaneously. This term englobes such a big number of different related practices that even Vander Wal has problems to make the initial meaning of the word to prevail [VW05].

A folksonomy can be defined as a collection of a set of users, set of tags and set of resources, and a ternary relation between these three types within a time dimension [DS08]. As a difference of traditional taxonomies, folksonomies do not explicitly define relations between the terms. Folksonomies have no hierarchies and, as the resources are tagged by the users, they are a representation of the users' vocabulary [Qui05]. The users of a system define the meaning of the terms in the folksonomy through their individual choices at the time to decide which tags appropriately describe the resources [GLYH].

Folksonomies are generally divided into: *broad folksonomies*, that define structures that have been generated as a result of aggregating data from various users tagging the same resource, and *narrow folksonomies*, that define structures that have been generated as a result of aggregating data from individual users tagging their own resources [HKG+12].

In other words, tagging offers users the possibility to start categorizing contents according to their immediate needs without having to previously design a taxonomy. The result of this categorization is the folksonomy. In contrast, the construction, maintenance and enforcement of controlled vocabularies is often too expensive in terms of development time and requires the users to learn the classification schemes [GLYH].

2.1.2. Semantic and Cognitive Considerations

Many problems inherited from the natural language exist at the time to create semantic relations between words. These problems are caused by imperfections of the language, such as polysemy, synonymy and homonymy. Other problems arise from variations in the cognition among users. All these imperfections have an impact on the inconsitent usage of tags [GH05].

- **Polysemy**: The term polysemy comes from *poly* (more than one) and *semy* (meaning), and denotes all those words that have multiple meanings or that are applicable in similar yet not equal contexts, e.g *pole* might mean the *mast* of a boat, a long piece of wood or metal used as support or a walking stick for blind people. In tagging systems, polysemy is a problem because it dilutes the results of the queries returning related terms to the introduced one, but that might make no sense in the working context.

- **Homonymy**: The term homonymy comes from *hom* (same) and *onymy* (name), and denotes all those words that have the same spelling but differ in meaning, e.g *pole* might refer to a post or mast, any of the two locations on earth where its axis of rotation ends (North and South poles) or also to make reference to people native from Poland. Homonymy is strongly related to polysemy but it has to be noted that, while the former phenomena appears as a result of the evolution of different word roots into the same form, the latter is the application of the same word root into similar contexts. It is important to note that these phenomena are not exclusive, *pole* is an example of polysemic and homonymic word. In tagging systems, homonymy is way easier to solve than polysemy, because those can be largely ruled out in a tag-based search with the help of related terms that help to indicate the unwanted homonyms, thus not being shown in the interfaces [GH05].

- **Synonymy**: The term synonymy comes from *syn* (with/together) and *onymy* (name), and denotes all those words that share similar if not identical meanings, e.g *car* means the same as *automobile*. This is the greatest of the problems of this nature for tagging systems, because it is difficult for taggers to be consistent in the terms they use for tags. In collaborative systems, where taggers might have different cultural backgrounds this problem is emphasized.

- **Cognitive categorization**: The level of specificity at the time to describe a resource also plays an important role in tagging. Even though it would be expected of taggers to use different levels of specificity, it has been proven that some patterns exist. Users tend to go for the denominated *basic level* of specificity, like for example with the use of *bmw* instead of *vehicle* (superordinate level) or *6 series coupe* (subordinate level) [GH05].

- **Other considerations**: For example, plurals, upper-cases, whitespaces, hyphens, or spelling variations, such as *dependent* and *dependant* or *organisation* and *organization*, might also influence the efficience of the system.

2.2. Social Tagging

Social tagging, or also called collaborative tagging, is the term coined as a result of the use of tagging to categorize and navigate through resources in the framework of a community of users. Tagging to organize digital content is not new, but it has been with the advent of the Web 2.0, and the apparition of web-based tagging systems such as Delicious and Flickr, that it has gained popularity.

The use of tagging in document repositories or digital libraries allows a categorization of the resources by simply assigning keywords. Nonetheless, such organizing tasks have been traditionally performed by authorities, like librarians or experts in the matter. As opposed, social tagging involves the idea of allowing everyone to freely attach such text labels to resources. This approach is especially useful for those collections whose size outgrows the capability of single authorities to do the categorization [GH05]. It also applies in the absence of authorities. Both facts happen at the same time in the web, reasons that easily explain its success and rapid expansion in this context. Despite the rapid growth of popularity of these systems, relatively few studies have been performed on the matter [MNBD06].

Web-based tagging systems, such as the already mentioned Delicious and Flickr, allow users to publicly tag and share content, so that they can organize their and others' resources for their own sake while browsing resources through categories built by the rest of the community. Thus, social tagging involves both individual and collective implications. However, social tagging systems exist in a wide variety of concepts and architectural designs that imply notable differences. For example, some systems might only allow users to tag their own resources [GH05].

Social tagging systems also have the potential to improve search, retrieval and exploration of resources, while they offer the possibility to detect spam, create reputation systems, mine data, and introduce new modalities of social communication [MNBD06].

On the other hand, current social tagging systems have important flaws like very limited navigation options, since no explicit relations between tags are established [MNS12a]. Moreover, the use of tagging collaboratively stresses the complications derived from the natural language such as the vocabulary problem, synonyms, homonyms or other of the commented in section 2.1.2.

In this chapter, we will provide an overall summarization of the main implications of social tagging. In *The tagging three-part model* we discuss about the architectural decisions that might be taken considering the users, tags and resources of the system, and how these might influence the final *folksonomy*. Finally, we present some of the most prominent web-based approaches of social tagging, with a special attention to Delicious and Flickr.

2.2.1. The tagging three-part model

A unified user-tag-resource model is useful for search and information retrieval, information organization, discovery and communication, spam filtering, reducing effects of link spam, improvement in trust on metrics, identification of trends and globally or community emerging topics, and to locate experts and opinion leaders in certain specific domains [MNBD06]. The model in Figure 2.1 represents the implicit relationship between resources through the users that tag them or similarly, users are connected by the resources they have tagged.

Figure 2.1.: *The tagging three-part model involves users, tags and resources. [Smi08].*

- **Users**: Users are the main actors of the model. They utilize the system attaching text labels to the different resources of the system to fulfill any of their possible interests, needs, goals or motivations. These can be to organize for easier future retrieval, to share resources with other users, etcetera. It is also important to consider that in some systems, the resources may as well have been added by them. In other words, users are not necessarily tagging their own resources (See Section 2.2.1.2).
- **Tags**: Tags are the keywords attached by the users to the resources in the system. Tags are any strings of information users find valuable to have attached to a certain resource, and they entirely depend on the user. It also depends on the user at the time to interpret them. Tags can be classified in several types, such as personal or public, indicating location, use, time, etcetera. As users differ so do their tags: *"You Are What You Tag"* [HHH08]. Tagging allows and even encourages these differences (See Section 2.2.1.3).
- **Resources**: Resources is the term that describes all those items that are tagged by users. Everything uniquely identifiable in our world is taggable, and therefore, a potential resource. Nowadays, the most popular resources happen to be bookmarks, photos, videos, books, films, etcetera. The idea is that within a tagging system, all the resources share something in common or are of the same nature (See Section 2.2.2).

The simplicity and flexibility of tagging requires for some complex design decisions to be taken. The architecture of tagging systems is denoted by a set of rules that define the interaction between users, tags and resources. These rules intrinsically involve trade-offs, for example the introduction of social components influences the way in which people

contribute and have a significant influence on the kind of tags they introduce. What is clear is that differences between tagging systems influence significantly on the resultant tags. Also, the different personal and social motivations that prompt individuals to collaborate affect the system itself [MNBD06].

In the following sections we will present the most relevant observed attributes of system designs, user motivations and tag types, which have a major influence on the utility and usefulness of the social tagging systems and how users appropriately use them to discover or retrieve resources, or to socially interact, if not all of them together [MNBD06].

2.2.1.1. Dimensions of Tagging

The content and usefulness of the tags of a system greatly depend on various design decisions and attributes concerning the following key dimensions of tagging systems [MNBD06]:

- **Tagging Rights**: The most important attribute of a tagging system design is the definition of who can tag which resources. These *rights* offer multiple possibilities and combinations. The most obvious are *self-tagging*, where users are only allowed to edit the tags from their own resources (e.g Technorati), and *free-for-all tagging*, where everyone can tag any resource in the system (e.g Delicious). However, these are not unique. Other more advanced approaches allow users to specify different levels of permissions on who can tag their resources (e.g Flickr with the different contact type groups). Distinctions between who can add and who can remove tags might also apply in some case. For example, at the time to remove a tag it can be interesting to distinguish cases where noone, everyone, the tag creator or the resource owner can do it. The less restrictive the system is, the more rich and complex the data is. It is easy to understand that resources tagged in *free-for-all* systems will comprehend a bigger and broader amount of tags than those which are individually tagged. For example, tags assigned to a photo may incorporate different points of view depending on if the tagger is the author of the picture, a professional photographer, a friend or a stranger. Finally, the visibility of the tags should also be considered. For instance, Pinboard appeared as an alternative to Delicious for users who were not interested in sharing their resources and tags. Nonetheless, the more restrictive, the less social.
- **Tagging Support**: The way tags are introduced in the system has a great impact on users behaviour and cooperation. The three main categories observed in already working systems are: *blind tagging*, where users tag resources without any indications from the system (e.g early versions of Delicious), *viewable tagging*, where users can see the current resource's tags (e.g current versions of Delicious), and *suggestive tagging*, where users get tag suggestions from the system. These suggestions might be based on other tags previously used by the user himself, tags previously added to such resource by other users or tags added to related resources by other users inferred by the system. Tag suggestions have an important impact on both tagging and searching for resources. Suggestive systems help to consolidate tags within a system, reducing, for example, concept divergences inherited from language, such as synonyms or singular versus plural. Artifacts such tag suggesters enhance the

convergence of folksonomies, hetereogenizing the pool of tags in a system and reducing the chaos produced by those tags, one of the biggest concerns of tagging. However, it is not yet concluded if such consolidation is necessarily a good thing. For example, mistakes or imprecions might grow at a pace where a big part of the community takes them for granted, and thus, becoming truth or illegitimately right. It also must be taken into consideration that for both, *suggestive* and *viewable tagging*, the first taggers of the system, or of each concrete resource, will have a great impact on the behaviour of the following taggers.

- **Aggregation**: System designs can be categorized depending on if the multiplicity of tags for a certain resource is allowed or not. When allowed, it is called *bag-model*, and otherwise, *set-model*. The former, allows creating frequency statistics for given resources, permitting to infer the taggers' collective opinions, or relationships between them, between tags or between resources (e.g Delicious). The latter permits users to collectively tag resources avoiding repetitions (e.g YouTube).

- **Resource types**: The types of resources being tagged also play an important role on tagging behaviours. Typical resources nowadays are bookmarks, books, videos, music, events, bibliographic material, photographies, podcasts, goals, blogs, etcetera. Any virtually representable object can be tagged or used in a tagging system. Although implications deduced from this dimension are yet to be empirically tested, hypotheses point out differences between textual resources, such as books, and non-textual resources, such as photos. Also the amount of tags attached to a resource of a type might be way greater to that of another type, just because much more can be said. For example, a movie could be tagged by its director, screenplay writer, cast, and other people who made it possible, while a bookmark tends to be limited to the main uses or information contained on the link.

- **Source of material**: Another important distinction is to consider who provides the resources. These can be supplied by the system users (e.g YouTube, Flickr, Delicious) or by the system (e.g Last.fm). Restrictions on this aspect might come from technological limitations, especially for free services, (e.g Flickr, YouTube) or through social norms (e.g CiteULike).

- **Resource connectivity**: Resources can be connected within the system independently of their tags. This connectivity is generally grouped in three groups: *linked*, where some resources point to other resources through the use of links, *grouped*, where resources are grouped in other concepts, such as Flickr groups, and *none*. These connections might be used for folksonomy convergence purposes, such as to provide meaningful tag suggestions.

- **Social connectivity**: The same way resources can be connected within the system, some systems might allow user connections. The same categorization applies. User connections might be symmetric, where both share the same connection, or asymmetric, for example, the "follow" connections. The most relevant implication of social connectivity is probably the apparition of localized folksonomies based on the social structure of the system. Two different groups of people might be tagging a set of resources differently depending on their cultural background, localization, etc.

2.2.1.2. User Motivations

Users' motivations also greatly affect the nature of the tags in social tagging systems. These motivations can be both, personal needs or sociable interests, with purposes that vary from contribution to the collective process of tagging to adaptation of that to their own needs [SKK10]. Users' behaviour is significantly affected by the taken model decisions despite they are unaware of that, thus varying the underlying motivations among users and systems. Motivations are usually divided into two main practices: *organizational* and *social*. The former arising as an alternative to structured organization systems and the latter, as a social communication tool [MNBD06].

Motivations have been categorized by several authors and are not necessarily exclusive from one another. Below, we provide the summarization realized by [GLYH] (based in [MNBD06] among others):

- **Future retrieval**: Users tag resources to ease future retrievals of these, both for themselves and others. Users might tag a collection of resources with an special label to group them to ease their future group search. These tags can also act as reminders, for example, *totag* indicates which resources need to be tagged in the future. These descriptive tags provide very useful metadata about resources that still need to be tagged.
- **Contribution and sharing**: Tags that describe resources and conceptually attach them to clusters or refined categories are also widely used. These type of tags help to define relations between resources and tags and facilitate the sharing with either known or unknown audiences.
- **Attract attention**: Tags might also be exploited to draw attention to one's resources with the aim to cause a greater effect on the social community. Normally, respresentations of tags and its relations in the interface, such as *tagclouds*, incentivize users to contribute aiming at a personalization of the global view and the most popular tags. Those might also lead to one of the biggest problems in tagging: spam [KSHS09].
- **Play and competition**: Tags are introduced to the system based on internal or external sets of rules, such as when social groups develop their own rules to engage in the system, by seeking particular features and categorize them. Some users might try to alter the system for fun, such as getting meaningful sentences or paragraphs out of tagclouds or other visual tag representations.
- **Self presentation**: Also called *self-referential tags*, appear from the necessity to achieve notoriety in the system. Some users might tag resources with their own real or fictional identities as a way of leaving a mark in certain resources. A good example of this is the *seenlive* tag in Last.fm, where users marked resources, in this case events, related to them, in this case, events they had attended.
- **Opinion expression**: Tags can be used to express opinions on the resources, so that other users can get feedback or realize the general opinions regarding such resource. For example, *hilarious* can denote a very funny video in YouTube.

- **Task organization**: Tags can be utilized to organize tasks. A very common example is the *toread* tag, that denotes resources that still need to be read. These tags can be used to, for example, group resources needed to develop a certain task, such as grouping papers on tagging for the development of this thesis.
- **Social signalling**: Users might also want to tag resources to communicate information about the object to others. For example *brokenlink* is used in Delicious to mark those bookmarks that are not working anymore.
- **Economic compensation**: Users might be economically compensated for categorizing resources within a system. Some sites like Squidoo enhance the use of tagging in this manner.
- **Technological ease**: Some users might tag because the technology they use to upload resources eases this practice.

These wide variety of motivations enhances the apparition of distinct roles among the users in social tagging systems. Similarly to the roles that emerged in Wikipedia for article contributors, correctors, categorizers, etc, in social tagging, roles such as describers, categorizers, publishers, leaders, tag heterogenizers, etc, apply [TSMM08]. Since the apparition of social tagging, many studies have categorized the different roles users tend to perform within the communities.

2.2.1.3. Kinds of Tags

As a result of the different architectural tagging dimensions and user motivations many different kinds of tags appear. The existence of such types in every tagging system strongly depends on the decisions taken at the time to design it and they do not necessarily need to be present in every single system.

A possible categorization of tag types is presented below [GLYH] (widely based on the 7 tag types for social bookmarking in Delicious proposed by [GH05]):

- **Content-based tags**: Tags that identify the actual content of the resource. These tags tend to be descriptive and generally extrinsic to the tagger, so one can expect significant overlap among individuals. Some examples can be *Germany*, *politics*, *airplanes*, *fjord*, *java*, etcetera.
- **Context-based tags**: Tags that identify the context of the resource in which it was created or saved. These tags also tend to be relatively independent of the user that attaches them making the probability for many people to use the same tags very high. Typical examples are those related to place and time, such as *Munich*, *November*, though other contexts apply, like for example *olympics* or *Oktoberfest*.
- **Attribute tags**: Tags that are inherent attributes of a resource but may not be directly derived from its content. These tags typically apply to the identification of characteristics or qualities of the resource. For example, *hilarious*, *inspirational* or *innovative*.

- **Ownership tags**: Tags that identify who owns or who created the resource. Identifying the resources' ownership can be particularly important for example in Flickr photos, where professional photographers need to mark the authorship of such work.
- **Subjective tags**: Tags that help users express and share their emotions and opinions towards the resource. For example, *overrated*, *amazing* or *mademecry* apply. This kind of tags can be successfully used to generate recommendations, as they involve subjective item qualities, and they enhance the so called *opinion mining* [PFO+08].
- **Organizational tags**: Tags that help to the personal organization of users, such as task reminders, like for example *totag* or *toread*. This kind of tags tend to be personal or depend greatly on the creators interpretation, making these tags useless for global aggregation and summarization. It is also important to consider that these tags are not perennial, tags like *toread* will disappear from a resource as soon as it is read.
- **Purpose tags**: Tags that denote goals or purposes users expect to achieve with the use of such resources. Typical tags are *tutorial*, tolearn, or more specific tags like *learnLaTeX*, *translator* or *pastetool* [War].
- **Factual tags**: Tags that identify facts about the resources. These tags are the most widely applied and they are strongly related to the first three listed types: content-based, context-based and attribute tags. Therefore, these also tend to be extrinsic to the taggers.
- **Personal Tags**: Tags that are only used by the tagger himself. Normally, they are used to organize user's resources in groups, such as, own resources, self-reference resources or resources related to task organization. Some examples are *triptokorea*, *myguitarsolos* or *mymastersthesis*.
- **Self-referential tags**: Tags that refer to themselves. A good example of this is the *sometaithurts* tag of Flickr, that groups a collection of images taken from Flickr or of people using Flickr.
- **Tag bundles**: Tag bundles make reference to the idea of tagging tags. These practice develops into the creation of hierarchical folksonomies and they aim to improve the navigability through the tags of the system. Examples of this can be *programminglanguages* or *computerrelated*.

As we have seen, some of the types are of an extrinsic to the user nature, facilitating the overlapping of tags among different taggers. On the other hand, there are also types where the information that the tags provide is only relevant to the tagger who has added them. Therefore, some tags are used by the majority of the people for a specific resource, while others might be used by few or just one person, depending on the level of specificity. Those tags used by more people tend to be more meaningful at a global scale [GH05]. Finally, it has been observed that users initially tend to use general tags, being the first tags used the ones with greater frequency within that resource, and the following decreasing in popularity. From such observations, it can be concluded that the first attached tags to a resource represent the basic concepts, because they are not only widespreadly agreed on, but also are the first terms that users thought of at the time to associate them with the resource [GH05].

2.2.2. Examples

Tagging systems have become an indispensable tool in the framework of Web 2.0. These, are applied to a wide variety of resource types that require different architectural designs. Some relevant examples are: *LibraryThing.com*, a social book cataloging site where users and experts tag books simultaneously [HGM09], *YouTube*, a video sharing system that allows videos to be tagged, *CiteULike*, a bibliographic references catalogue, *Technorati* and *tumblr*, blogs where users are only allowed to tag their own resources, *Last.fm*, where both songs and events are tagged, *amazon*, an e-commerce company allowing users to tag all their products, and so on and so forth. Nevertheless, the ultimate paradigms for social tagging are *Delicious* and *Flickr*, which are presented below.

2.2.2.1. Delicious

Delicious is a social bookmarking service launched by Joshua Schachter in 2003, being the the first relevant example of social tagging. *Social bookmarking* is the specific application of social tagging with bookmarks as resources. Other examples of social bookmarking sites are *StumbleUpon*, a web application that suggests links depending on users' preferences, and the already mentioned *CiteULike* [WZB].

Delicious basically allows users to tag their links, URLs of whatever they find interesting on the web, for future retrieval and also to share them with the rest of the community. It therefore includes a powerful tool for discovering bookmarks. The three main design principles that make this sharing and discovery possible are the use of a tagging system, its openness, that is the possibility for everyone to tag everyone's resources, and the inclusion of a subscription system (typically known as *"follow"* users option) [Lee06].

One of the finest features offered by Delicious is the aggregation of everyone's resources by tag. For example `http://www.delicious.com/tag/wikipedia` lists all the bookmarks any user cared to tag with the label *wikipedia*. Furthermore, several tags can be used as filters for this aggregation, e.g */wikipedia+catala* would show all those resources concerning the catalan wikipedia. Finally, the subscription network allows users to engage more easily in sharing and discovering tasks, as users can *follow* other users with similar interests [Lee06].

2.2.2.2. Flickr

Flickr is a social media service that allows users to upload photographs, view photos shot by other users in the system, and basically offers all the general features of Web 2.0 to enhance users participation [LJ06]. One of these, are the tags: Flickr allows users to tag their images with descriptive keywords. Not only that, but Flickr also advocates the use of tags as a way to improve search of the user's own images and to improve the findability of such pictures for other members of the community. Sharing images with other users through groups and increasing the visibility of their images in general is very important for the users, and tags help in this matter [LJ06].

2.3. Tag Presentation Techniques

One of the main problems tagging systems encounter is the difficulty to provide effective tag presentation techniques that guarantee easy navigation options through the tags of the system. These difficulties arise from the poor information on relationships between tags raw *folksonomies* incorporate. These define no explicit links between tags, so that tags can only be related by the frequency in which they occur together in the resources [MNS12a].

As a consequence of these limitations, *search* is the main feature using tags. Normally, current systems provide few related tags to the tag being browsed which only enables users to jump from one tag to another easily changing contexts [MC09].

In this chapter we present some of the most significant approaches to minimize the mentioned difficulty. These are of varied nature but few patterns can be observed. For example, both *tag-clouds* and *tag-clusters* try to minimize the navigation problem from raw *folksonomies*, in other words, without requiring users to provide additional data, while *tag-hierarchies* and *hierarchical faceted categories* require users to perform minimal further tasks. Of course, combinations of these techniques also apply, like for example the *TMine* [SCDCS08] and the *TagFlake* [DCSCS08] which merge hierarchies with tagclouds or the blending of facets into tagclouds ([TLP+12] and [SvZ10]'s *TagExplorer*). Furthermore, alternative approaches like for example those that enhance tags by weighting them have appeared [LHY+09].

Independently of the approach, some general rules apply at the time to present tags to the users. For example, to show tags in an alphabetical order helps users find information more easily and quickly. Other chracteristics such as the position and distribution of the tags within the interface, or design decisions such as font size and colour also play a big role in the success of the tag presentation techniques [HK07].

2.3.1. TagClouds

TagClouds are a visual representation of a set of words, in our context a set of tags, in which text attributes such as size, weight or colour are used to represent relevance differences among the words of the represented set [RGMM07]. The set of tags is selected by certain rules, dependant on the architectural design decisions, normally combining available data from tag co-occurrences, resources and users. Several algorithms to choose such sets of tags exist, with the goal to improve the quality of the sets regarding their actual relation to the context in which the user is at every moment [MC09].

TagClouds are becoming an increasingly featured tool in social tagging sites, where content is categorized through always evolving *folksonomies*. For example, they appear in all the commented sites in Section 2.2.2, in one way or another [RGMM07].

A typical pattern that can be observed in the various versions of tagclouds is to map the prevalence of a term within the set of tags using bigger fonts and its recentness using brighter colours. Other typical enhancements are the so called *spatial algorithms* which try to pack and redistribute the selected tags so that the final tagcloud requires smaller

areas within the interface. Furthermore, the use of *clustering algorithms* focuses on placing tags with similar meanings or that are frequently used together one next to each other [RGMM07].

TagClouds support navigation as they are automatically generated tables-of-contents or indexes for sets of information resources. Moreover, similarly as traditional glossaries do for books, tagclouds provide means for users to form a general impression of the content of a website, or part of one. For example, in social tagging sites where tagclouds can be generated from the relations between tags and users, tagclouds can provide a pretty accurate overview of a person, showing her interests and expertise [RGMM07]. Although these facts have been proven, the widely held belief that tagclouds are useful for navigating social tagging systems has been debated [HTSA11].

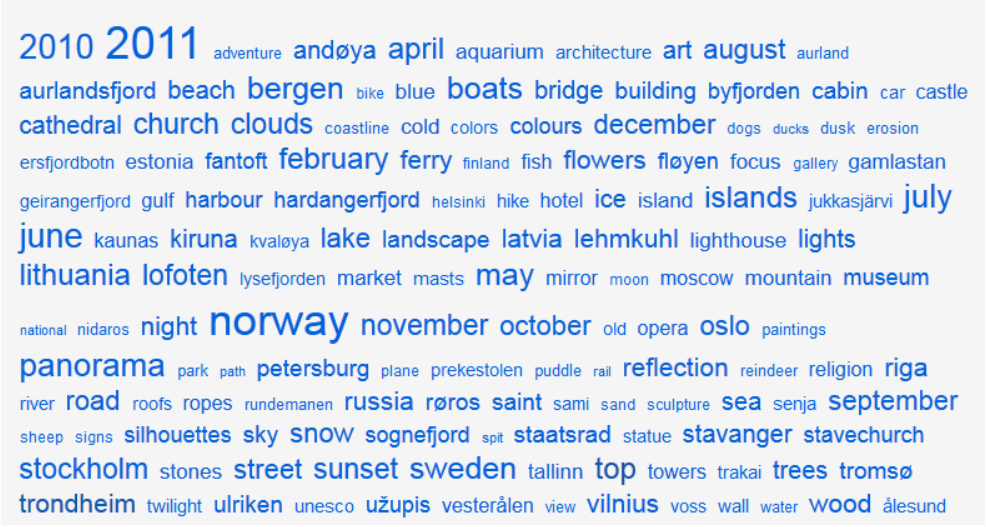


Figure 2.2.: *An example of TagCloud as presented on Flickr.*

Depending on implementation decisions, tagclouds can represent sets of tags for different purposes. Some of the most relevant tasks tagclouds can support are [RGMM07]:

- **Search**: Finding specific or similar concepts is relatively easy with a tagcloud. If tags in the tagcloud are related with topics users are desiring to browse, they will feel they are on the right track, thus feeling more comfortable with the interface. It is important to note, though, that traditional search interfaces are normally preferred to tagclouds for this purpose, which leads to the conclusion that tagclouds are not sufficient to search in a *folksonomy*-based dataset [SCH07].
- **Exploration**: Tagclouds are very useful when it comes to browsing through a set of tags without a special target or topic in mind. Thus, they are especially appropriate for exploratory searches and discovery.
- **Impression formation**: Getting a general impression on the main keywords related to a resource or person is easily achieved with the aid of tagclouds. This fact is true, even considering that users generally ignore a great deal of the tagclouds' tags.

- **Recognition**: Tagclouds have also proven helpful at the time to identify resources or people. For example, tags can easily help to distinguish which is the Max Mustermann who is interested in music festivals from the one interested in philosophy.

The success of tagclouds is also evident by the great amount of written literature on both, tagclouds as means of navigation through tags ([HMHS06], [TSHA] and [HTSA]) and how to improve their interfaces ([LZT09]).

Nonetheless, tagclouds have important limitations that need to be dealt with. The inevitable noise and redundancies from uncontrolled tags, intrinsic problems of the free nature of tagging, represents a big problem for tagclouds. Moreover, the selection of tags to be shown only by the common usage frequency might turn into a problem of high semantic density, which means that very few different topics are shown and prominent tags dominate the cloud excluding those that are less important [CSBT07]. Finally, tagclouds provide a poor representation of the structure and relation between the shown tags [HMHS06].

2.3.2. TagClusters

TagClusters appear as a result of the application of *clustering* algorithms, which look for co-occurrences of two tags and calculate the probability of them appearing together. Therefore, tagclusters are basically an algorithmic approach to group tags being semantically related. Depending on how frequently tags appear together among resources and users who have applied them, different relationships can be inferred.

Clustering techniques, have proven to be one of the best ways to learn about the relationships between tags, and thus, multiple different clustering algorithm approaches have been proposed [KSS10]. These algorithms strongly depend on the system's design, and are also thought to suit such designs emphasizing their particular characteristics.

	0	1	2	3	4
0	apple osx mac	software freeware windows	howto ubuntu linux	java performance	tech computer hardware
1	programming python _net development	tips	security		
2	3d graphics opensource			audio	

Figure 2.3.: *An excerpt of an example of a clustered tagcloud. The complete example can be found here* `http://nlp.uned.es/social-tagging/asonam2009/tagcloud/`.

Data clustering can also be understood as a statistical data analysis technique used to find strongly related tags. This way, clustering achieves to partition the main pool of tags of a dataset into subsets of similar objects, or *clusters* [BKS06].

A wide variety of styles for tagclusters' visual representation have been proposed, such as *clustered tagclouds* which try to mimic the behaviour of tagclouds but grouping subsets of the shown tags (See Figure 2.3) [ZGPFM07], or those based on *overlapper*, a visualization tool focused on the connections and overlappings in data (See Figure 2.4) [ST06].

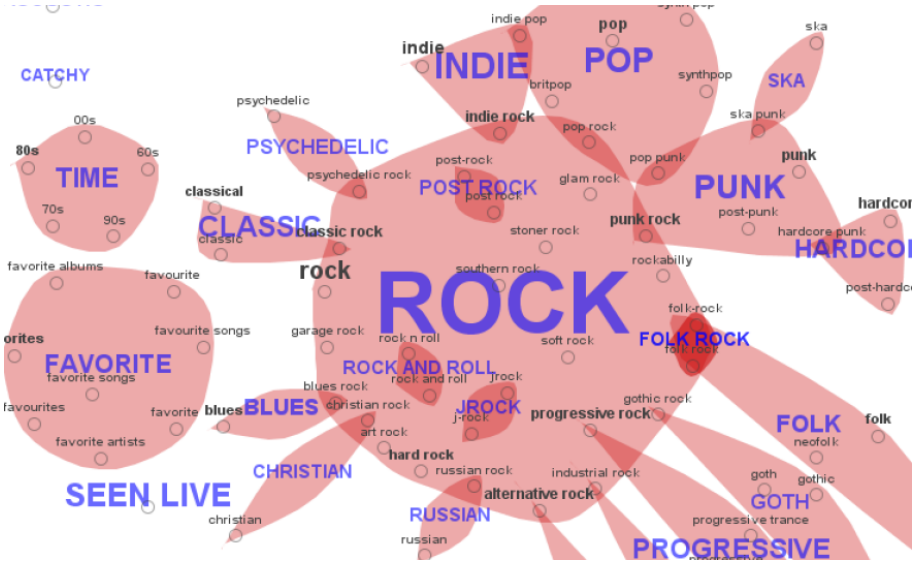


Figure 2.4.: *An example of the visualization overlapper approach for tagclusters [CSBT07].*

However, the best known visual representation of clusters is probably the *clustery goodness* proposed by Flickr, which, provided a popular tag, shows related tags grouped into clusters. For example, in Figure 2.5, provided the tag *eagle*, the returned clusters clearly fall into two semantic categories of eagles: the animal and the fighter airplane.

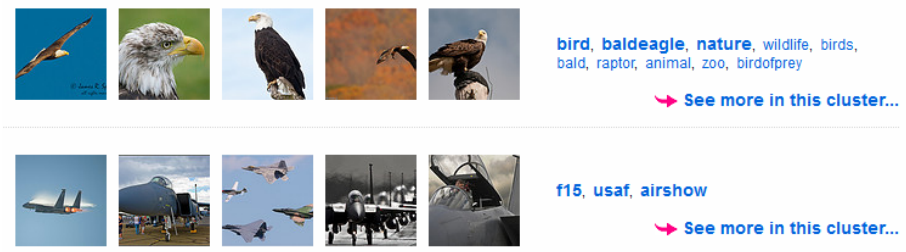


Figure 2.5.: *An example of the visualization of clusters as shown in Flickr. In this case, two clusters are shown for the tag "eagle".*

2.3.3. Hierarchical Faceted Categories

Hierarchical Faceted Categories (HFC) are a category system that organizes a meaningful set of labels so that they reflect the concepts relevant to the domain. Normally, they are created manually by experts or end users while the assignment of documents to the categories can be automated to a certain accuracy degree. Hierarchical Faceted Categories have the advantage of being coherent and relatively complete, something that cannot be said of the unpredictable results of clustering algorithms. This has been proven by multiple studies where participants also prefer categories over clusters [Hea06a].

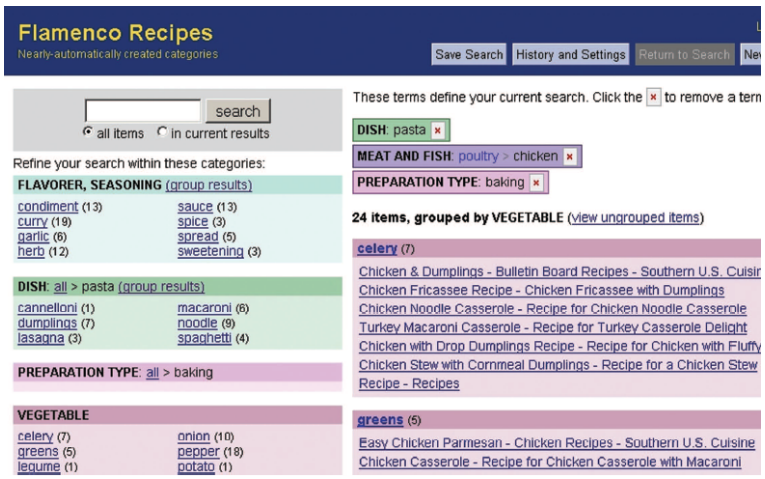


Figure 2.6.: *Flamenco: an example of hierarchical faceted categories [Hea06a]. Demos of these approach can be found on* `http://flamenco.berkeley.edu/`.

Hierarchical Faceted Categories represent a more complex approach compared to tagclouds and tagclusters and, therefore, the difficulty to create a usable and effective interface for exploration and browsing of information collections is higher. In [YSLH03] one of the most influential interface designs tackling this problem is presented. Figure 2.6 shows an example of hierarchical faceted categories presenting the different concepts applicable to a collection of culinary recipes. The idea for the design is quite simple. Instead of creating a unique large category hierarchy, a set of category hierarchies is displayed each of which corresponds to a different facet representing the different dimensions relevant to the collection being browsed. For example, in the context of the tag *chicken*, the various hierarchical faceted categories can be [Hea06a]:

- **Dish type**: *main*, *soup*, *salad*, *side*, *dessert*, etc.
- **Ingredients**: *meat*, *vegetables*, *grains*, *spices*, *salt*, *walnuts*, etc.
- **Cooking methods**: *bake*, *fry*, *grill*, *boil*, *braised*, *poached*, etc.
- **Cuisinte type**: *italian*, *indian*, *french*, *spanish*, *pakistani*, etc.
- **Dish style**: *schnitzel*, *tandoori*, *sautéed*, *marsala*, *stroganoff*, etc.

As we can see, each of these facets comprehends a hierarchy of terms associated with it.

Also, during the navigation through these facets, more than one tag of these hierarchies can be chosen. In other words, the structure and design in which the tags are shown does not constrain their use, without giving up on useful properties of tagging such as its inherent flexibility.

Interfaces such as the one proposed in Figure 2.6 allow flexible ways to navigate through the contents of the collections. One can access various levels of specificity for a certain hierarchical facet, for example, accessing the subcategories of *poultry* and then *chicken*, and later choosing a category of a different facet such as *schnitzel*, providing a list of recipes containing both tags, like for example *chicken schnitzel "Münchner Art"* or *chicken schnitzel "Wiener Art"*. Interfaces for these technique generally include previews of where to go next and tools for users not to lose their bearings within the system, so that they can easily return to previous states without confusion [Hea06a]. One of the most used such tools are the attribute *breadcrumbs* [Ins02]. Normally, the free text search feature is also included.

Hierarchical Faceted Categories reduce mental work of the end users by promoting recognition over recall with the use of meaningful suggestions. Also it has been proven that such organization structures provide a powerful platform for exploration and discovery [EHS+01]. However, a trade-off between the usefulness of the system and its learnability exists. Diverse studies have proven that such structures are relatively harder to learn but preferred within a short time than standard keyword-and-results listing interfaces used in Web search engines. These studies also conclude that with such designs it is less likely to be lead into dead ends [Hea06b].

Fagan presents a summarization of the main positive findings that support the appropriateness and convenience to include faceted classifications in information retrieval interfaces [Fag10]:

- Facets are useful for the creation of navigation structures.
- Faceted categorization is more efficient for retrieval in database searching.
- Facets help to avoid dead ends.
- Users can achieve a higher level of efficiency when using faceted systems.
- Finding relevant results is easier and faster in faceted systems.
- Despite initial discrepancies users tend to prefer facets at long-term.
- Users prefer search results organized into predictable hierarchies.
- Users are more confident when using faceted systems.
- Unfamiliarity with the innovative designs is a problem at the time to begin using the interfaces for faceted systems.

However, hierarchical faceted categories also bear relevant drawbacks in contrast to the tagcloud and tagcluster techniques. Here, categories of interest need to be known in advance, and to the detriment of important trends in data, these might not be shown. Furthermore, the greatest problem is that the hierarchies generally need to be created manually, which supposes an extra organizational effort [Hea06a].

3. Usability Evaluation

After having read about the tagging systems in Chapter 2, the importance of how these are presented to the users seems obvious and the necessity for an intuitive and usable interface a must. It is at this point where usability, and more concretely, usability evaluation, plays an important role on the development and subsequent success of the user interface. In our case, the TACKO tagging system user interface.

However, usability evaluation can also be ineffective and even harmful if naively applied. Usability evaluation should not be done *"by rule"* but *"by thought"* [GB08], and this is something we will have to consider at the beginning of our evaluation.

For such consideration, we will need the necessary knowledge. Therefore, in this Chapter 3 we will provide the basic background knowledge about usability and usability evaluation that we will be using for the development and analysis of the proposed interface for TACKO in the Chapters 4, 5 and 6.

3.1. Usability

User interfaces have now way more importance in software development than they used to. It's been demonstrated that, in average, almost half of the written code in modern software is dedicated to enhance the user interface [MR92]. Therefore it is reasonable to dedicate a big amount of the effort to ensure the usability of these user interfaces.

As the importance of usability has been proven, we find it necessary to get a better idea of what it really is and the main concepts it comprehends. We provide such information in the section "Definition". Secondly, in "Attributes of Usability" we will briefly describe the main components of usability. We will talk about the relations between these attributes in "Usability Trade-Offs". Finally, as one of the two more important issues in usability is the individual characteristics of the users [Nie93, p. 43], we find appropriate to dedicate a few words to the categories of users and their individual differences.

3.1.1. Definition

ISO, the International Standard Organization, defines usability in the international standard for the evaluation of software quality (ISO 9126) as *"A set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users."* [ISO01]. Alternatively, in the ISO 9241 they define usability as *"the effectiveness,*

efficiency and satisfaction with which specified users achieve specified goals in particular environments" [ISO10], where:

- **Effectiveness** is the accuracy and completeness with which specified users can achieve specified goals in particular environments.
- **Efficiency** is the resources expended in relation to the accuracy and completeness of goals achieved.
- **Satisfaction** is the comfort and acceptability of the work system to its users and other people affected by its use.

A less detailed definition and generally accepted can be *"the ease of use and acceptability of a system for a particular class of users carrying out specific tasks in a specific environment"* [Hol05]. The term ease of use refers to the performance of the users, how fast can they start using the system successfully, and their satisfaction towards it. On the other hand, the term acceptability refers to the actual success of the system from the point of view of if users actually use it. Obviously, both terms are strongly related. The easier it is for a user to learn and understand a system, the more satisfied she will be, and thus, the greater possibility that the user will finally use the product.

Usability can also be considered as a very specific issue, subcategory of the larger "system acceptability", which is concerned on *"whether the system is good enough to satisfy all the needs and requirements of the users and other potential stakeholders, such as the users' clients and manager"*, in the model of attributes of system acceptability proposed by Nielsen [Nie93, p. 24-25]. Following this model, usability can be understood as a quality attribute. There are many other important quality attributes. A key one is utility, which refers to the design's functionality: Does it do what users need? So, in other words, while utility concerns on whether the functionality of the system can do what is needed, usability concerns on the difficulty users suffer to use such functionality. Usability and utility are equally important: It matters little that something is easy if it is not what users want. On the other hand, it is also not good if the system can hypothetically do what users want, but noone can successfully make it happen because the user interface is too difficult. To study the utility of an interface, the same user oriented evaluation methods that improve usability presented in the second part of this chapter can be applied [Bev95].

It is also important to note, that usability applies to every single aspect of a system with which a human interacts. That includes, for example, installation and maintenance procedures. There are very few occasions in which a computer feature has no user interface components. In conclusion, usability shouldn't be understood as uniquely related to advanced and complex user interfaces, but to every single computer-human interaction, independently of how simple it might be [Nie93, p. 25].

3.1.2. Attributes of Usability

Usability is not a single property of the user interface and therefore, can be decomposed into several components or attributes. The most generally accepted classification of these usability characteristics is the one proposed by Nielsen which consists of: learnability, efficiency, memorability, errors and satisfaction [Nie93, p. 26].

It is thanks to these more precise terms in which we have decomposed usability, that make it an engineering discipline. Usability is no longer just a matter of discussion but it can be now systematically approached, improved and evaluated like any other discipline [Nie93, p. 26-27].

We now provide a deeper description of these attributes:

- **Learnability**: the system must be easy to learn so that users can begin successfully working with the system as soon as possible.

  This attribute is the essence of usability: Every system needs to be learnt in order to be used, and learning it is the first process of the interaction. Without a succesful understanding of how the system works, this can not be adequately used.

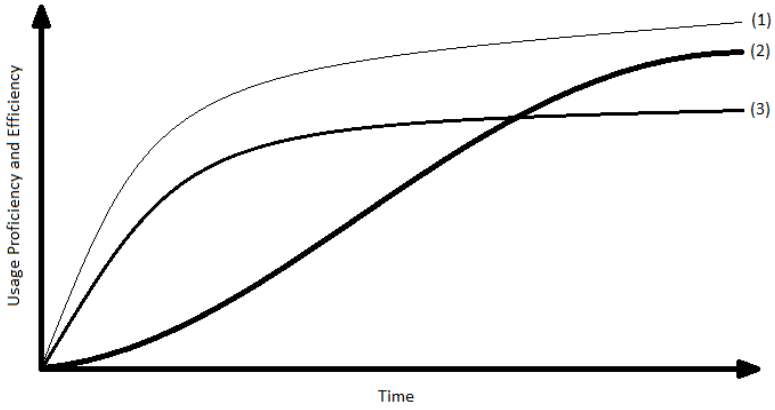


Figure 3.1.: *(1) Ideal learnability curve. (2) Learnability curve that focuses on expert users. Initially hard to learn system but finally more efficient. (3) Learnability curve that focuses on novice users. Initially easy to learn system but finally less efficient.*

Nielsen uses the representation in Figure 3.1 to denote the learning curves for both, novice and expert users on a specific system. The initial steepness of the curve shows how easy a system is to learn. The steeper the easier it is to learn. This representation helps the analyst to decide whether he puts more emphasis on this attribute or in others such as efficiency. For a better explanation of the trade-offs between these attributes see section "Usability Trade-Offs".

Normally it is supposed that when users start using a system they are totally unable to use it, being represented at the (0,0) position in the coordinates, but it is also important to note that this is not always the case. For example, this doesn't apply

to interfaces that are based on other interfaces that the users have used before. We are not only referring to improved interfaces for the same applications, where the users just need to transfer their skills into the new supposedly improved interface, but also to interfaces that use features the users might already be familiar with. For example, the text editor we are using to write this document (notepad++), includes a tab system that lets you navigate through different text files and it was not difficult to understand because we are already familiar with the tabs offered by most of the modern web browsers.

Learnability is together with satisfaction the most measurable component of usability. For example, it can be measured in time, how much time does a user need to master the system, in number of clicks, or in any other metrics [GFA09]. The tasks a user is able to complete successfully, also in specified time limits, will help the analyst to categorize how apt a user is towards the system, and therefore, measure how learnable the latter is.

Finally, it is always useful to remember that users never completely learn how an interface works before actually using it. They learn by doing and that means they normally know only what they need to know. Therefore, it is important to make a distinction between complete mastery and sufficient level of proficiency achieved by the users when it comes to do useful work [Nie93, p. 30].

- **Efficiency**: the capability for a user who has learnt the system to attain a high level of productivity.

  Efficiency is represented as a variable in the x-axis of the Figure 3.1. Considering the user already knows the system, what is the maximum level of productivity she can achieve? Normally, users keep learning indefinately, in some cases at such slow paces that one can believe they got to their tops, that they have learnt everything they need, ”enough” [CR87].

  Efficiency is measured on experienced users. As defining a user as experienced is somewhat subjective, a definition of expertise appears to be crucial, in such a way that a group of users that match the defined level of expertise can be chosen. The group representing a definite level of expertise will be then tested accordingly, measuring the time they need to complete specific tasks [Nie93, p. 31].

- **Memorability**: the system must be easy to remember so that the users who had previously used it, can now retake their tasks without a great effort.

  Nielsen defines ”casual users” as the group of users who can not be considered neither experts nor novice. Basically users who might have been experts in the past, or that had at least used the system before, and that, therefore, their level of expertise on the system should be superior to that of the novice users. This group is marked as the third biggest group of users. Thus, the importance of developing an interface whose functioning is easy to recall.

  Memorability is strongly related to learnability. Normally, improvements made due to evaluation on the latter have a great impact on the former.

Although memorability is less important and consequently less thoroughly tested than other attributes, it has two main recognized measure techniques. The first one consists of measuring the amount of time a casual user needs to achieve tasks he was familiar with in the past, so that it can be compared to the time he actually needed when he was considered an expert. The second one consists of memory tests immediately conducted after the usage of the system, where users can be asked for how to develop certain tasks in detail, for example asking which icons he needed to click.

However, memorability testing has proven to be, in some cases, misleading, or unimportant. Systems have found alternative methods to substitute problems caused by bad memorability to an extent that users might be unable to remember how certain parts of the system look like, but still use them successfully the next time.

- **Errors**: the system should have a low error rate, so that users make few errors while using the system, and those few errors are still easily undone. This meaning that unsolvable mistakes are unacceptable.

  An error, is normally defined as an action that does not successfully achieve the desired goal. The error rate of the system is then based on how many of these actions are there in the total overall of actions executed by a user at the time to conclude a task. These error rates are also used as a measure for other usability attributes [Nie93, p. 32].

  Errors can be grouped in two main categories: Those that are easily detected and normally corrected by the user, which are already analyzed in terms of efficiency, and those that are more problematic, either because the users do not detect them and that leads them to incorrect outcomes or because they are hard to undo, or plainly irreversible.

- **Satisfaction**: the user likes the system because it is pleasant to use, and she has the feeling it helps her to successfully perform the tasks with it.

  This attribute is especially important for nonwork environments, such as home computing, games, interactive fiction, etcetera, because in such environments, the users give more value to the entertainment they are getting out of the system. That is their unique goal.

  The most common method to measure satisfaction is by simply questioning the users for their opinion and experience on the system. Their independent answers are obviously too subjective but the overall average can clearly provide an idea of how satisified are the users in general.

  Questionnaires provide the necessary consistence to the study of this attribute. It is especially important that users do not answer such questionnaires until they have already tested the system as it has been demonstrated that answers dramatically change if users try the system with the questions in mind [RD83].

  Another interesting observation is that users normally tend to relate the level of difficulty of a system with the ”peak difficulty” they experienced rather than with the

"mean difficulty" [Nie93, p. 33]. A user will always recall a system's difficulty by the greatest problem they found while using the system. In other words, it will be enough for users to rate a system as more difficult that another one to find an action that it is more difficult to perform than all the available actions in the second system. An obvious conclusion is that one can not rely on user ratings when it comes to evaluate the system.

Questionnaires regarding this attribute of usability tend to be short and exist many varied approaches. For example, in the "Likert scale" users would indicate their degree of agreement with each statement on a scale from 1 to 5, being 5 that she totally agrees. On the other hand, "semantic differential scale" questionnaires would show two opposite terms and expect the user to mark the proximity or distance from each of the terms he agrees on, following the idea that the more you agree on something, the more you disagree on its opposite. Such methods incredibly help to detect inconsistencies and get to better conclusions, for example questioning the same targets from different points of view.

There are trade-offs between these attributes, some that are more relevant than others, depending basically on the situation and the specific usability goals for the project. For example, one of the most common is sacrificing rapid learnability in favour of a better final attainable efficiency [SP04]. We will talk more deeply about these trade-offs in the next section.

3.1.3. Usability Trade-Offs

Figure 3.1 can easily be misinterpreted. One can believe that if a system is easier to learn, then the highest reachable efficiency level is inferior, and otherwise, if it is finally more efficient it is probably harder to learn. Thankfully, this is often not true. Generally, systems that are quickly understood by novices have also a good impact on experts.

This is generally achieved by providing multiple ways to perform a specific task. This different possibilities to perform a task will be of diverse difficulty, and this difficulty is normally related to the level of efficience attainable by using such method. Novice users will normally start using simple and slower features to end using later in the future the so called "accelerators", *"elements that allow the user to perform frequent tasks quickly"* [Nie93, p. 41]. A good example of an accelerator is the *tab* key, which will let the user navigate through the different fields of a form without using the mouse. Another example are the shortcuts. Unfortunately, having multiple ways of performing a task adds complexity to the interface and therefore it is important to design it in a smart way. For example, novice users should not have to face the advanced features, or, in other words, those should not be that easy to find.

However, offering multiple ways to complete a task is not the only solution to solve the learnability-efficiency trade-off. There are cases in which the same features can be used, providing a bigger amount of information for the novice users while the experts can easily ignore such tips. For example, filling in text fields by default.

In spite of all the techniques to maximize the optimization of the different usability attributes, one has to keep in mind that it is impossible to achieve the ideal score at all of them at the same time. *"Trade-offs are inherent in any design process and apply no less to user interface design"* [Nie93, p. 42]. A good example is the interest on avoiding irreversible errors. Decisions to counter this problem, like confirmation alerts, might reduce the efficiency of the system and the performance of the users.

3.1.4. Categories of Users

The revolution in personal computers, now accessible for almost everyone due to the decreasing hardware prices, are making computers available to ever broader groups of users. These groups of users, utilize those for a large and new variety of tasks that might have never been considered before, while in the past, computers were used by more specialized, and thus smaller groups of users, who were experts in their fields and performed very specialized tasks. Therefore, the demand for usability is growing rapidly in the last years [Nie93, p. 8].

Besides the users' task, their individual characteristics and differences are one of the main issues of usability. Hence, it is important for usability engineers to know the users and to be able to categorize them in groups.

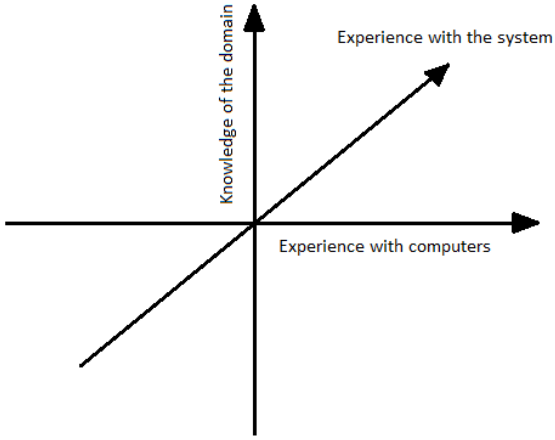


Figure 3.2.: *The three differing main dimensions on users' experience.*

In Figure 3.2, the three main dimensions on which users' experience differ are represented: knowledge about computers in general, understanding of the task domain and expertise in using the specific system. Being the latter the dimension normally referred to when talking about the "user expertise". This dimension has a great impact on the decisions made on the development of the user interface. It is not the same to design an interface intended to be used by novice users than one thought for experts [Nie93, p. 44]. It is obvious that the considerations taken for a one-use interface, such as the interactive screens one can find in a museum, are going to be different to those thought for an advanced

tool. However, distinguishing between novice and expert users is a very undefinite way to describe the real world. There are no such users. Normally, most of the users will not master all the parts of the system, but only those they need to, in a way that a user considered an expert, can be a novice for determinate features of the system. Another way of presenting it would be that users never reach the ideal level of expertise and therefore, they are always dependant on the help initially thought for beginners.

It is also very important to consider the general knowledge about computers users have before they start using a specific system. Some of the features of our system can work, or should be developed, based on similar features of other systems that have successfully been applied before. In a way, users expertise on our system will not be starting from 0, and this will provide enough confidence to navigate through more innovative features. One can easily think of similarities between modern systems, for example, most of the applications dealing with text incorporate the "ctrl+c" "ctrl+v" hotkeys to an extent that users even take for granted such feature will exist in new similar systems. In fact, not providing general knowledge features would be in some cases a fatal stepback for our interface.

Furthermore, the user's knowledge of the task domain of the system is also very relevant. This normally makes reference to the level of specialization the interface should integrate. If the users are not expected to be familiar with the domain, then it makes absolutely no sense to use very advanced terminology, or in the cases where it is vital, then the interface has to keep in mind to provide the necessary tools to guide this users to a good end.

Distinguishing users by their experience is very useful, but it is not the only factor one should consider. Other factors that might be relevant are age, gender, learning style, reasoning abilities, etcetera [Nie93, p. 46]. Differences in these factors and how they are treated in our interface will have great impact in the individual performance of the users. It has been demonstrated that an individual whose performance might be one of the best in a concrete interface, might turn to be one of the worst in a different interface that values more other details, and viceversa. Motivation or attitude towards the system is also a decisive factor and thus, the importance of boosting the satisfaction users perceive when using it.

In conclusion, the diversity of individual users and groups of users should not mislead us at the time to achieve our goal, which consists on developing an interface where every single user fits in to a greater or lesser extent. Development based on individual feedback is not always the best idea, as users tend to ask for features to personalize the interface and suit their individual preferences.

3.2. Web Usability

Now that we have a better general idea of what usability is, and considering that the interface for our tagging system, TACKO, is going to be a web interface, we find interesting to provide more information on usability in the web: Web Usability. This is one of the biggest branches of usability and has become very popular due to the boost of the world wide web.

Jakob Nielsen is, again, the reference on the subject especially because of his book "Designing Web Usability", and his definition of Web Usability is the most accepted: *"Web usability is an approach to make web sites easy to use for an end-user, without the requirement that any specialized training be undertaken."*

In this chapter we will present some of the main considerations and conclusions Nielsen presents in his book, which will be of great importance for the tests and improvements proposed for our system in chapters 5 and 6.

3.2.1. Main Considerations

The evaluation of the usability of our web should help us determine how to present the information to the user in a clear and concise way. For this purpose it is important to decide the proper distribution of the features of our web application so that we can provide the users with everything they might need, but also in a way that is not totally new for them. Trying to avoid ambiguities is also a very important task. These very basic guidelines are the key to get the users' attention, of vital importance for the success of our web application. As Nielsen says, *"the mouse is in the hands of the user and he decides everything"*. The fact is that in the world wide web it is incredibly easy to go anywhere else, the rest of the world's competitors are just one click away [Nie99, p. 9]. The competition is very high and you are competing for the time and attention of the users against many other sites. The web is an "economy of the users' attention", where the currency is the time of the users [Nie99, p. 160]. The key to success are the loyal users, those that always come back [Nie99, p. 380].

The content is the attention focus of the users. It is the real reason for which they come to visit us. Therefore, the quality of the content and the capability to find it are the two main criteria for usability [Nie99, p. 160]. The content is so important that it should use between 50% and an 80% of the screen, being the rest for the navigation, etcetera [Nie99, p. 22]. Something that seems obvious but many people ignore is that you should not create false expectations on the users. Offering pages or features that are not yet available is one of the main causes of frustration among the users, and this will definately have a negative impact on you interface [Nie99, p. 164].

The use of stylesheets is totally recommended but these should come together with a brief explanation so that not only the developer understands them [Nie99, p. 83]. As general recommendations for these stylesheets we can find: no use of more than two different font

styles, use relative and not absolute sizes and use the same name for the classes representing the same concepts in all the stylesheets [Nie99, p. 84-85].

It is always better to design interfaces that are easy to learn without any documentation, but in the cases where this can not be avoided, the explanations must be very practical. Normally simple and unambiguous examples are enough [Nie99, p. 129].

The error messages must be constructive and help the users solve the problems they might find. To merely inform the user there has been an error, or say which error was it, is not enough [Nie99, p. 111]. Also it is recommended to offer previsualizations of the multimedia content. If the user has no real idea of what she is going to see, it is very probable that she will not take the time to see it [Nie99, p. 134]. Animations should be avoided [Nie99, p. 143].

Many designers tend to ignore usability considerations and design how they want, or what is worse, to please the boss, instead of trying to satisfy the user [Nie99, p. 13]. Every single site should provide the main features a user needs in the simplest way possible [Nie99, p. 380]. Another very important criteria to satisfy the users, and that despite the improvement of the internet connections still has to be beared in mind, is the response time. This must be one of the main criteria for the development of every web application [Nie99, p. 42].

The welcome screens or *"splash screens"* should disappear. These just slow down the user experience, making them have to click more times to get to the main page, which is what they normally want to see [Nie99, p. 176]. Offering buttons or links that will take the user to where she already is is also a bad idea [Nie99, p. 166]. The navigation must help the users to get answers to fundamental questions such as: Where am I? Where do I come from? Where can I go? [Nie99, p. 188]. As commented before, users do not like changes. Breakthrough changes are normally unwelcomed. It is very easy for the user to find common features in the usual positions such as the logo at the top left side, the search bar at the top right and the main navigation structure in the left column [Nie99, p. 213].

Regarding to the search tools, it is important that they are available from everywhere in the site. In very specific cases special search tools also apply [Nie99, p. 225]. The bigger the search bar is, the bigger amount of words will the user type and therefore get more precise search results [Nie99, p. 233]. URL preservation is also important. Many users learn URL in order to access faster, and these can also help the user get oriented inside the application [Nie99, p. 249].

3.2.2. Considerations on the users

In order to get the best of our interface we need to know the users. It has been demonstrated that the main reasons for users to come back are high quality content, constant updating, minimal response time, ease of use and if the users' needs are fullfilled.

The users are very impatient, and they demand instant answers to their actions [Nie99, p. 10]. If your application does not respond quickly it is likely that the users will leave. If the response takes less than a second it gives the feeling of freedom through navigation.

0,1 seconds are considered as an instantaneous response. Approximately 3 seconds are the limit to keep the users' attention. Over that limit the user finds something new to do while she waits for the browser to load and it is important to inform her about the progress, like for example with the so called progress bars or simple messages. There are many factors that slow the responses between the server and the browser but the users' only perceive the sum of all of them to react negatively or to reduce the confidence on the site they are visiting [Nie99, p. 44-45]. Finally, it is important to note that pages over 1.5MB suffer around 25% of user disconnections because they are not willing to wait.

Only 10% of actual users use outdated software and hardware, but ignoring this 10% of potential users is not a good decision [Nie99, p. 97]. The web has to be oriented to all publics. Pages that are focused on the most advanced users are not successful because these type of users are not the majority [Nie99, p. 364]. The updating of browsers to the newer versions is done at a pace of a weekly 1%, which means that 2 years are needed for absolutely all the users to be using the update [Nie99, p. 34]. Furthermore, new software takes at least a year since its first beta version to be broadly used. Using the last technical advances is often not recommended because users will need to download it and install it when visiting the page. The majority will not stay [Nie99, p. 131].

Normally, when users' visit a page for the first time their sight goes for the content, they check the titles and subtitles so that they get an idea of what the content is about and it's only then, that they either decide to read the content or search for the navigation features to find other pages [Nie99, p. 100]. It has been demonstrated that reading from the screen is more tiring and a 25% slower than from traditional platforms [Nie99, p. 106]. The 79% of the users first examine a page and then start reading it [Nie99, p. 104]. No user is devout enough of a site so that he will make big efforts to learn how to use it [Nie99, p. 196]. Users do not read documentation nor manuals voluntarily. They do it when they really need to [Nie99, p. 129]. The abuse in animations, especially for marketing purposes, has made users to totally ignore information that moves. They just do not pay any attention to it, which contrasts with the idea this animations were thought for, which intended to get the attention using the movement [Nie99, p. 146]. Another very interesting statistic is that if a user finds a problem in the main page, she will only end up finding the page she was initially intending to find in 42% of the cases. For the development of more advanced tasks this percentage can get down to 26% [Nie99, p. 164].

For the users, the navigation is a load they have to deal with. Therefore, it is important not to abuse it, and offer only the needed navigation features at each point [Nie99, p. 18]. The 16% of users' abilities in the use of a site are based in the standards for the colors used in the links. All the other criteria together represents the other 84% [Nie99, p. 64]. The use of the browsers' "back" button is the second most used tool of the browsers, after the link buttons. It is an essential element for the users to feel safe [Nie99, p. 86]. Also, half of the users browse mostly doing searches, a fifth prefer to do it clicking on the links and the other 30% combine both behaviours [Nie99, p. 224]. Users detest to use the scroll, so it is recommended the pages to be short [Nie99, p. 103]. They will scroll for a long article in which they are interested but to navigate, they will always use options that are on their screens [Nie99, p. 115]. Horizontal scroll is generally hated and its use should be totally avoided [Nie99, p. 175]. Users tend to complain when a site uses very different navigation

interfaces different from those they are used to [Nie99, p. 189]. Frequently, when the navigation is designed, it is assumed that the users will get to where they expected directly, but this is not always the case [Nie99, p. 195]. One of the most frequent feelings of the users is that they are not in the page they expected to be.

Finally, it has been observed that users rarely add more than one word in small input fields [Nie99, p. 233] and the majority of searches done in websites contain two words [Nie99, p. 237].

3.2.3. The eight golden rules of interface design

After getting to know better how users interact with the interfaces it makes sense to try to get some conclusions out of the observations. This is something Ben Shneiderman successfully sums up in his book *"Designing the user interface"*, where he proposes a collection of principles that derive from his experience and are applicable to most of the interactive systems after being properly interpreted, refined and extended [SP04, p. 75]. This collection, comprehending 8 principles are widely known as *"the eight golden rules of interface design"* and, despite their simplicity entails some limitations, they are a good starting point for both students and expert designers. This list of principles is strongly related to the *"Ten Usability Heuristics"* proposed by Jakob Nielsen. The eight rules are [SP04, p. 74-75]:

1. **Strive for consistency**: Consistent sequences of actions should be required in similar situations. That means that all the prompts, menus and help screens should use identical terminology and consistent color, layout, capitalization, font, etcetera. If exceptions are required, then this should be very limited in number and easily comprehensible for the users.

2. **Cater to universal usability**: The needs of diverse users must be taken into consideration. Design for the diverse categories of users providing the necessary features for novices, such as explanations, while giving the freedom to more advanced users to increase the pace of interaction through the use of shortcuts, abbreviations or macros, that will improve the perceived system quality.

3. **Offer informative feedback**: For every user action, the user should get feedback from the system. For those actions that are minor and frequent simple feedback is enough, but for those that are not that frequent or apply major changes, a substantial feedback is encouraged. Changes can be explictly shown in the visual presentation of the objects.

4. **Design dialogs to yield closure**: Sequences of actions should be organized into groups with a beginning, middle and an end. Understanding such actions as groups enhances the idea of progress, and when users get the informative feedback at the completion of one of such groups, it gives them the feeling of accomplishment, a sense of relief, and it serves as an indicator that the way for the next group of actions is clear.

5. **Prevent errors**: The system should be designed in such a way that users cannot make serious errors. If a user makes an error, the interface should detect the mistake

and offer a simple, constructive and specific solution for recovery. Mistaken actions should not apply any changes to the system, or at least, the interface should provide the necessary instructions to restore the previous non-erroneous state. It is important to offer simple error handling.

6. **Permit easy reversal of actions**: The system should be designed so that the actions are easily reversible. The possibility to undo actions, whether they were mistaken or not, will relieve the anxiety of the user towards the interface, as she will know that mistaken actions can be taken back. Moreover, it encourages the exploration of unfamiliar options or features of the system. The operations to reverse actions might range from single actions to data-entry tasks or groups of actions.
7. **Support internal locus of control**: Expert users like to have the feeling they are in charge of the system and that the system responds to their actions. Unfamiliar interfaces or inability to execute the desired actions cause anxiety and dissatisfaction to the users. The system should be designed so that the users are the initiators of actions rather than responders to the system.
8. **Reduce short-term memory load**: Human capabilities to process short-term memory information are very limited. Therefore it is strongly recommended that the displays are kept simple, multiple page displays are merged, window-motion frequency is reduced and to provide the users with enough time to learn the suquences of actions, shortcuts, and other more advanced features.

3.3. Usability Evaluation Methods

The application of the design principles for usable applications is not enough to ensure the usability of the final product. Even though accurate design techniques are applied, it is still necessary to check both, the intermediate and final results, to verify if it effectively provides the required features and meets the user requirements. It is at this point where evaluation, whose role is to help verifying such issues, becomes important.

The main goals of evaluation are [DFAB98]:

- Assess the application's functionality
- Verify the effect of its interface on the user
- Identify any specific problem with the application, like for example, unexpected behaviours of its features

Concretely, evaluating web applications consists of verifying if the interface design of the application easily allows users to retrieve and browse the contents, and to perform several tasks using the offered functions as well. Thus, it is not only important to appropriately present the contents and services available within the application, but also to make them easily reachable for the users [MRT].

Evaluation is normally divided into *formative* or *summative* depending on the phase in which it is performed. The former takes place during the design while the latter takes place after the product has been developed, including this when a prototype version is ready [MRT]. The goal of the formative evaluation is to check if the team of designers is aware of and understands the users's requirements during the early stages of the product development. It also focuses on testing the design choices quickly and informally, so that the team of designers can get initial, and normally significant feedback. On the other hand, the goal of summative evaluation is to support the detection of users' difficulties at the time to use the interface, and to qualitatively improve the product with future upgrades [Sch].

There are different evaluation methods within these two broad categories that can be applied at different stages of the product development. Therefore, it is important that the developers know the different usability methods to be able to determine which methods are best suited at every stage of their project. Independently of the project, it is greatly recommended that usability is considered before the prototyping takes place [Hol05]. If an interface is not evaluated in early stages, it will easily turn out unusable or of poor usability which will require changes in latter versions that might translate into high expenses and implementations of great difficutly. *"The earlier critical design flaws are detected, the more likely they can be corrected"* [Hol05].

Usability evaluation methods (UEM) are widely divided into two main groups: *Inspection methods*, which are evaluations conducted by specialists and end users play no role, and *test methods*, where the real end users are studied and evaluated. In the following sections we will describe in more detail some of the most applied methods of each of these two categories.

3.3.1. Usability Inspection Methods

Usability inspection refers to a set of cost-effective evaluation techniques or methods, that evolved from former code inspection methods used in Software Engineering for debugging and improving code, where evaluators examine usability related aspects of an application, trying to detect violations of established usability principles [Nie95], and then provide feedback to the designers about possible design improvements. The inspectors can be usability specialists or also designers or engineers with special expertise (f.e. knowledge of a specific domain). In any case, the application of such methods relies on a good understanding of the usability principles [NM94].

This category of methods appeared when the issue of cost effectiveness started having an important role in the field of usability evaluation [MRT]. As a consequence, many usability evaluation methods based on the involvement of specialists, to supplement or even replace direct user testing, were proposed [NM94].

Normally, usability inspection methods aim at finding usability problems in a design, but some may also measure the severity of the found problems and of the overall usability of the system [Nie95]. Inspection methods are suitable for early evaluations of the system's usability [Nie95].

In this section we will present the main inspection methods as presented in [NM94].

3.3.1.1. Heuristic Evaluation

Heuristic Evaluation (HE) is the most informal method and consists of having usability specialists judge if the different elements of the interface satisfy the established usability principles or heuristics [NM94].

Normally, it would be expected that the inspectors conduct their evaluations following a certain list of rules of recognized guidelines, but, as these guidelines tend to be extremely long and specific, most developers prefer to execute their evaluations based on their own experience, intuition and common sense [Nie93, p. 155]. Thus, making it more informal.

Theoretically, heuristic evaluation is a systematic evaluation of a user interface design for usability [NM94], whose goal is to find the usability problems, so that they can be appropriately fixed, using a set of heuristics like the *"Ten Usability Heuristics"* proposed by Nielsen himself [NM94] (refined from [MN90]) :

- **Visibility of system status**: The interface should always keep users informed about what is going on, with the use of appropriate feedback and within reasonable time.
- **Match between system and the real world**: The interface should speak the users' language, using words, phrases and concepts that are familiar to the user. Thus, avoiding the use of system-oriented terms. Information should be presented in a natural and logical order.
- **User control and freedom**: Users make mistakes, and it is by making them that they learn to use the system. Therefore, we need to enhance undo and redo tools, so that

they can promptly recover from their mistakes. This will give them a feeling of safety that will have a great impact on the learnability of the system.

- **Consistency and standards**: The use of the interface elements should be consistent and have the same functionality as much as possible. Similar actions should be executed similarly. Follow platform conventions.
- **Error prevention**: Better than good error messages is a design that prevents users from making mistakes. Prevention is better than cure. If error-prone actions cannot be eliminated nor replaced make sure the user agrees to proceed by confirmation messages.
- **Recognition rather than recall**: The users should not have to remember information from one part of the interface to another. The instructions for the use of the system should be always visible.
- **Flexibility and efficiency of use**: Accelerators (presented in section 3.1.3), which are unseen by the novice users, often speed up the interaction for the expert users making, therefore, that the system is both usable for expert and novice users. Allow users to customize frequent actions, for example with the use of shortcuts.
- **Aesthetic and minimalist design**: The interface should never show irrelevant or rarely needed information. Every single information detail provided in the interface diminishes the relative visibility of other possibly more relevant elements, and may deviate the users' attention of the really important sources of information.
- **Help users recognize, diagnose and recover from errors**: Error messages should precisely indicates the problems in plain language (avoiding technicalities) and constructively suggest a solution.
- **Help and documentation**: It is always better when a system can be used without documentation or help. In cases where this is unavoidable, such documentation or help should be always provided, easily retrievable, focused on the user's task and list steps to carry out the tasks in a simple and concise way.

In [Nei09] we can find examples of how modern web applications apply these heuristics. Other sets of heuristics that are widely used are, for example, Ben Shneiderman's *"The eight golden rules of interface design"* presented in the section 3.2.3 of this document [SP04], or Bruce Tognazzini's *"First principles of interaction design"*, though this list might be slightly too long for an heuristic evaluation and might serve better as a useful checklist [Tog03].

Experience from several projects indicate that, despite an individual evaluator can perform an heuristic evaluation of a user interface, she will miss a great deal of the usability problems that it incorporates [Nie92]. Single evaluators tend to find approximately around 35% of the usability problems [MN90]. Furthermore, different evaluators tend to find different usability issues, making it considerably enriching to aggregate the evaluations of different evaluators [Nie93, p. 156]. This effect, commonly known as *"The Evaluator Effect"*, exists for both novice and experienced evaluators, both simple and complex problems, both problem detection and severity assessment, independently of the complexity of the system [HJ03].

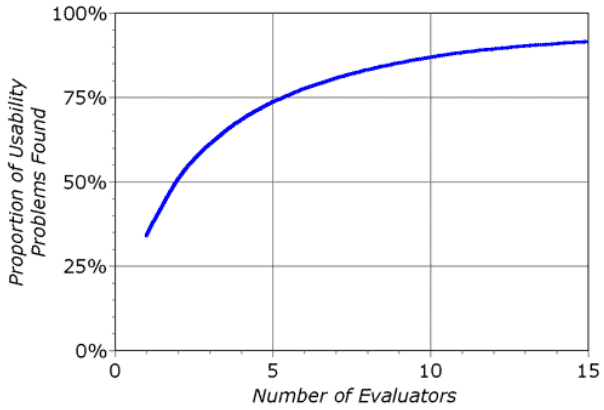


Figure 3.3.: *Relation between the number of evaluators using heuristic evaluation and the percentage of overall found usability issues [Nie93, p. 156]*

In Figure 3.3 we can see the relation between the number of evaluators using heuristic evaluation and the percentage of overall found usability issues. From this figure it is deducible that having more than one evaluator is greatly recommended and the minimum suitable number of evaluators should be between 3 and 5 [Hol05].

The initial idea is that every evaluator inspects the interface alone and, only after the evaluations have been completed, are the evaluators allowed to discuss their findings [Hol05]. This is important so that the different evaluators do not have any influence between each other. Of course, this does not apply for those heuristic guidelines intended for cooperative inspection.

Heuristic evaluation is especially valuable when time and resources are short, because skilled evaluators, without needing the involvement of representative users, can produce high quality results in a limited amount of time [KR97]. Heuristic evaluation has also proven to be a cheap and intuitive method for evaluating the user interface, especially early in the design process. Therefore, it was proposed as a substitute to user testing, which was considerably more expensive [HN07]. In heuristic evaluations with an observer, who is already familiar with the system and keeps track of the evaluators' observations, the responsability stays on the side of the evaluator, unlike in a user test situation [Nie93, p. 137]. Other differences to user testing are the willingness of the observer to answer questions from the testers during the session and the extent to which they will be helped through the use of the interface [Nie93, p. 157]. In user testing, the goal is to discover the mistakes users make, and therefore, the observers tend to be more reluctant when it comes to provide help [Nie93, p. 158]. Nonetheless, empirical testing has been proven to yield more severe problems than inspection methods [KCF92].

It is important to select the appropriate set of heuristics for the evaluated system as these can considerably diverge. For example, web-based sets contain additional heuristics that are only relevant to this field [Hol05].

Although this method was initially expected to be used by usability experts, it can also be

executed by less experienced evaluators with the expected decrease of efficiency [Nie93, p. 161].

Ideally, the result of an heuristic evaluation should be a list of usability problems in the interface related to those heuristics that may have been violated. Finally, these observations should provide enough information to develop a revised version of the design [Nie93, p. 159].

Advantages

- Finds many of the issues of the interface cheaply and early [HN07].
- Applies recognized and accepted principles [Hol05].
- Is intuitive. It doesn't require any special preparation to learn to use the method.
- Can be applied at early stages of development.
- Its application requires short time.
- It is the method that finds more issues in overall [JMWU91].

Disadvantages

- The level of expertise of the evaluators has a great impact on the final output.
- Several evaluators are needed. Generally it is not easy for developers to find experts [HN07].
- Most of the identified issues are minor to the detriment of more severe issues [KCF92].
- Some of the found issues may never bother users in actual use [HN07].
- Intrinsically entails a separation from end users [Hol05].
- Inspectors decide their own way to inspect. The evaluation of the entire design is not ensured [Hol05].
- No solution to the found issues is provided.
- The validity of Nielsen's guidelines has been questioned [GS98].
- Inspectors can misinterpret the heuristics, especially if those are too general.

Heuristic Estimation

Heuristic Estimation is a variant of Heuristic Evaluation (HE), in which the inspectors are asked to estimate and compare the usability of two or more designs in a variety of quantitative criteria, such as the expected user performance, as opposes to HE, where problems are merely identified [NM94].

3.3.1.2. Cognitive Walkthrough

The Cognitive Walkthrough (CW) is a usability inspection method that focuses on evaluating a design for ease of learning, particularly by exploration [WRLP94]. The fact that many users prefer to learn to use software by exploration, *"learn by doing"*, motivated the focus of this approach [WRLP94]. This causes that the cost of learning a new feature is determined by the immediate benefit such feature represents to the user [WRLP94].

The idea behind this method is that the analysts evaluate the system's functionalities in the context of one or more specific user tasks, simulating step by step how would users behave to perform every proposed task [WRLP94]. Thus, emphasizing cognitive issues such as learnability [Hol05].

An overview of the cognitive walkthrough process is presented below [WRLP94]:

1. **Define the inputs to the walkthrough**

   The input for a walkthrough session includes [WRLP94]:

   - **Interface's design description**: *"How is the interface defined?"* A detailed description of the interface's design might be in the form of a mockup or a working prototype.
   - **Task scenario**: *"What task(s) will be analyzed?"* An analysis of the target task(s) to be analyzed, both basic operations and combinations of those.
   - **Assumed user population**: *"Who will be the users of the system?"* A general description of the users in order to be able to consider their knowledge of the task and of the interface. An example can be "Flickr users who have previously worked with tags."
   - **Context of use**: *"In which context are the tasks going to be performed?"* A list of typical conditions under which the systems will be applied. For example, in a task where information must be retrieved from a database, it might be relevant to know its size.
   - **Sequence of actions**: *"What is the correct action sequence for each task and how is it described?"* A sequence of actions that a user should successfully perform to complete the designated task [WRLP94]. These descriptions are greatly conditioned by the level of expertise of the assumed population of users. For example, for advanced users "login to the system" might be enough, while for novice users a more detailed sequence must be applied, "Move cursor to upper top-right corner and click the 'Log in' button", "Introduce your credentials", etc.

2. **Convene the analysts**

   For group evaluations, the designer will present the design to a group of users, of varied backgrounds such as designers, software engineers, marketing analysts, documentalists or even better, interface evaluation specialists. Each person of the group will then carry out a specific role: one being the recorder or scribe, another acting

as a facilitator, and other worthy kinds of expertise such as knowledge of the potential market, user needs analyses, etcetera. All the members of the group are equally responsible for the execution of the evaluation [WRLP94].

3. **Walk through the action sequences**

   This phase comprehends the body of the analysis where evaluators try to mimic users' behaviour based on their assumed background knowledge and the goals they intend to achieve. For such task, the evaluators follow the problem-solving process proposed by Polson and Lewis (1990) which concludes that the 4 steps in problem-solving by the users are [HN07]:

   a) The user sets a goal.

   b) She determines the currently available operations.

   c) She executes the operation she thinks will take her closer to her goal.

   d) She evaluates the feedback given by the system.

4. **Record critical information**

   It is important to capture the information in an appropriate way so that the group can perform an effective evaluation. Sessions might be recorded, transcribed or kept in any other of the possibilities modern technology provides, thus, being very easy to go back and verify or retrace comments and decisions that might have been overlooked [WRLP94].

5. **Revise the interface**

   Finally, a revision of the interface is required to fix the issues that might have been encountered. There are four criteria in which to classify the issues [WRLP94]:

   - The user is not trying to do the right thing. Various solutions apply to this category: refactor the action, inform the user of which action must be performed, make the action consistent with the rest of the system so that the user realizes its necessity.
   - The user has the correct goals but does not know the action is available. The solution is to assign the action to a more obvious control, for example making it more visible.
   - The user does not know which action will achieve the effect she desires. The solution is to provide labels and descriptions that match the actions, including words users are likely to use when describing the development of the task that requires such action.
   - The user does not know that what he did was right and exactly what he wanted to do. The solution is to provide feedback.

At the time the cognitive walkthrough process was presented, the two main problems with CW were, first, that the task of filling out the forms was extremely tedious, and second, that the process found a very limited range of problems [WRLP94]. It was at this stage

and to minimize these limitations that new approaches were proposed, where the form-filling task should be divided among the group evaluators and the simple tasks should be evaluated first [HN07]. The Evaluator Effect also applies in CW [HJ03], though the evaluations developed by novice evaluators can still be effective [HN07].

The cognitive walkthrough has also been successfully applied, in combination with semantic analysis, in the evaluation of how well web applications support their users [BPKL02].

Advantages

- Independence from end users [Hol05].
- Independence from a fully functioning prototype. It can be applied at early stages. Can also inspect paper prototypes [Hol05].
- Helps designers to put themselves in the users' place [Hol05].
- Effective identification of problems that arise from interaction with the system [Hol05].
- Helps to define users' goals and assumptions [Hol05].
- Applicable by both, novice and expert evaluators [HN07].
- Supported by strong theories of learning by exploration [WRLP94].

Disadvantages

- Compiling all the data is a tedious work [HN07].
- The range of found problems is limited [HN07].
- Focuses only in one of usability's attributes: learnability [WRLP94].
- Evaluators may not behave as users would [WRLP94].

3.3.1.3. Pluralistic Walkthrough

The Pluralistic Walkthrough (PW) [Bia94] is an adaptation of the Cognitive Walkthrough that incorporates representative users to work together with the other expert analysts, such as designers, product developers, etcetera [HN07]. Thus, permitting that real users, system designers and possibly other experts, discuss on the tasks they all are evaluating. The representative users have to match with the system audience descriptions, as defined in the first phase of the walkthrough.

At early stages, documentation or help functions are rarely available and, normally, system designers will serve as *"living documentation"* answering the questions that users may come up with [Rii]. Thereby offering the users the necessary help to carry on with their tasks while the designers get valuable hints for their documentation. A usability expert administers the evaluation session, performing the role of moderator between users and designers. This task is important because its goal is that the designers' attitude towards

the users' comments to remain positive [Rii]. In the eventuality this were not to happen, users' willingness to give comments might vanish soon [Rii].

Pluralistic usability walkthrough is defined by the following five characteristics [Bia94]:

1. The method includes three types of participants: representative users, system designers and usability experts.
2. Screens are presented in the same order as they would appear to the user when using the system.
3. All the participants take the role of a user.
4. The participants write down the actions they, as users, would take to perform the given tasks.
5. The different approaches are shared and discussed. Firstly, the administrator presents a correct answer. Secondly, the representative users present their solutions. Finally, the system designers and usability experts present their solutions.

In Pluralistic Walkthrough, *"lucky guesses"* do not go unnoticed as the users can easily report that, although they had the right solution, they were not sure about it [Rii]. This approach is able to offer users' feedback even if the interface is not fully developed, thus, enabling to apply changes *"on-the-fly"* during early stages of the design [HN07].

This method grants even more evaluation perspectives, because of the differences in expertise of the evaluators, than those provided by the Cognitive Walkthrough. Thus, minimizing the *"Evaluator Effect"* [HJ03]. It has been demonstrated that perspective-based inspection methods grant that experts find, in average, a higher number of problems than those found when using heuristics [ZBS98].

In any case, the use of pluralistic usability walkthroughs is widespread even though it is not always referred as such. One can assume the pluralistic feature of the walkthrough might have become such a standard practice that is not mentioned anymore [HN07].

Advantages

- Greater number of usability issues can be found at once compared to CW.
- Applicable at early stages [Bia94].
- Applicable by both, novice and expert evaluators [HN07].
- Minimizes the *"Evaluator Effect"* [HJ03].
- Tracks *"lucky guesses"* [Rii].
- Possibility for the designers to get valuable hints for the system's documentation [Rii].

Disadvantages

- Does not address the usability issue of efficiency.
- Compiling all the data is a tedious work.

3.3.1.4. Feature Inspection

Feature Inspection is an inspection method that focuses on the set of features of a system, being each feature analyzed considering the main attributes of usability. Small subsets of features are evaluated individually so that they can be compared with other sets of features approaching the same goal in other systems. Feature checklists are used to compare the sets with the aim to show the differences between the similar tools. The output of feature inspection, helps, therefore, to get a better idea of how effective every set of features is compared to that of other similar approaches in the market [NM94].

Feature Inspection has a descriptive character. It checks for the availability of features rather than the absence of errors. Successful evaluations using this method depend mainly on the quality of the feature checklist and the expertise of the evaluator. Good feature checklists should incorporate the users' point of view related to what can actually technically be done. The most appropriate approach for developing such checklists is the top-down approach, where principal functions are evaluated before entering into more detail [NM94]. These checklists should constantly be updated with the arrival of new technical solutions. The evaluation should also be done following the same top-down order [NM94]. Both data analysis and reporting phases are reduced to a minimum, since the results barely allow any interpretation and the reporting is performed during the evaluation.

It is best used in the middle stages of development when the functions of the product are well known. Obviously it is suitable for products that are ready to be deployed. The output is generally represented by boolean values, normally representing the presence or abscence of features, which delivers highly objective results.

Advantages

- Emphasizes the importance of each functionality to achieve usability.
- Avoids evaluation of not so relevant features in favour of those sets of features that involve critical tasks.
- Does not require the execution of the evaluated program.
- Easily detects missing features.
- Tackles every usability attribute [NM94].
- Results can be generally considered objective (boolean results).

Disadvantages

- Not applicable at early stages.
- May overlook errors.
- Requires tremendous efforts in the preparation stage.
- The success of the evaluation depends considerably on the expertise of the evaluator.
- Certain features are hard to describe and also probably hard to find by the inspector.

3.3.1.5. Consistency Inspection

The main idea behind Consistency Inspection is to have a group of designers from multiple other projects inspect the interface in order to find out whether it approaches the different tasks in a similar way as their own designs [WJTC94]. The interface is systematically reviewed for its design's consistency, both within a screen and between screens, in graphics, text and interaction.

A Consistency Inspection should be conducted following these steps [Wil11]:

1. Create a small team of inspectors with different expertise backgrounds. Different perspectives will yield more consistency issues.
2. Define the target of the inspection, such as a certain screen or area of the product.
3. Define a list of which consistency attributes are the focus of the inspection, such as icons, menus, terminology, layouts, etcetera.
4. Provide the design against which you will compare the evaluated product.
5. Train the inspectors on:
    a) The definition of consistency and the different levels and attributes of consistency.
    b) Examples of consistency bugs so that they know what they are searching for.
    c) How to report the consistency issues they have found.
6. Provide the design of the product under evaluation, being it a wirefrime, a prototype or a working product.
7. Categorize and prioritize the found issues in order to carry on with the analysis in group.

Consistency Inspection standardizes how tasks are approached testing interfaces with others that have proven successful, thus, benefiting from attributes such as familiarity and memorability. On the other hand, it might be reluctant to the appearance of new approaches to perform certain tasks.

Advantages

- Ensures consistency within the project [WJTC94].
- Enhances the usability attribute of familiarity, benefiting from other similar approaches.
- Promotes consistency across groups of developers [Wil11].
- Applicable at early stages [Wil11].

Disadvantages

- Consistency has dimensions this method does not evaluate [Wil11].
- Might hamper innovative approaches.

3.3.1.6. Standards Inspection

Standards or Guidelines Inspection is a usability inspection method where an expert on specific interface standards or guidelines inspects an interface for compliance [NM94]. Basically, the inspector reviews the interface's design to determine whether this conforms to the chosen set of standards. These standards can be of a wide variety, such as industry standards, corporate standards, or standards specifically designed for the given project. Independently of their nature, the standards generally inspect issues such as screen layouts, fixed processes, response-time requirements, diplay configurations, terminology, spelling, etcetera [NM94]. The guidelines represent a summarization of good practice and provide useful guidance on the design of usable interfaces.

Advantages

- The guidelines help to ensure that the usability principles are tested.
- Increases the consistency between screens.
- The guidelines embody good practice experience in interface design.

Disadvantages

- May inhibit inspectors' capability to detect usability problems beyond the provided guidelines.

3.3.1.7. Formal Usability Inspection

Formal Usability Inspection is a method that consists of a formal review of the tasks a user completes while using the product [Gun95]. It involves stepping through the user's task, similarly to what is done in the pluralistic usability walkthrough but in a quicker, more thorough and technical way [HN07]. The main goal is to identify the maximum number of *concerns* in the interface as efficiently as possible. Its purpose is also to fix the found issues, thus, improving the ease of use of the product [Gun95]. Normally, the concept of *defect* or *problem* is substituted by *concern* with the intention to facilitate the acceptance of the process. Also, inspectors are encouraged to express these *concerns* in a constructive manner [Gun95] rather than simply criticizing the design.

The inspection team typically includes design engineers, usability engineers, customer support engineers, and when possible, customers. Normally, one of the experts performs the role of moderator [Gun95]. Formal Usability Inspections require information about the end users together with task scenarios, analogously to those provided with the pluralistic usability walkthrough [HN07].

The process of Formal Usability Inspection consists of the following six phases [KP94]:

1. **Planning**: The moderator helps the team of designers to form an inspection team, and to organize the inspection packet and meeting schedules. The inspection packet

contains information about the end users, the task scenarios and a description of the prototype or design.

2. **Kick-Off Meeting**: The inspectors are presented with the inspection packet and told what is expected from them. Normally, all doubts about the execution of the evaluation should be clarified at this point.

3. **Review**: Inspectors take the role of users, based on the provided information, and try to perform the task using the walkthrough methodology, keeping track of all those concerns they may find.

4. **Logging Meeting**: All the concerns are brought and discussed together. A log of all the discussion and concerns is kept.

5. **Rework**: The inspection team is gathered to try to find solutions to the found concerns, so that the product designers can review and fix their product.

6. **Follow-Up**: The moderator requests feedback on the process from the inspection team in order to generate a final inspection report.

Various companies in the early 1990s, such as Hewlett-Packard (HP)[Gun95] and Digital Equipment Corporation (DEC), proved the utility of these method and its capabilities to detect several usability concerns, finding even the small ones other methods may overlook [HN07]. Nowadays, the absence of literature indicating current use makes it difficult to conclude that it is as effective as other methods [HN07].

Advantages

- Improves the ease of use of the design [Gun95].
- Its application is faster than Pluralistic Walkthrough while preserving similar results [HN07].
- Design engineers learn how to evaluate their designs from the user perspective [Gun95].
- Applicable at early stages [Gun95].
- Its application is cheap and cost-effective [Gun95].
- The number of found usability concerns is high [Gun95].
- Increases awareness of user needs.

Disadvantages

- The expertise of the team of inspectors still has an impact [Gun95].
- The moderator has tedious work to do.

3.3.2. Usability Test Methods

Usability Test Methods are the set of usability methods that study and evaluate with real end users. Testing with real users is the most fundamental usability method and should always be applied, even if used as a complement to usability inspection methods [Nie93, p. 165]. In fact, this combination of both sets of methods is encouraged as they tend to report different types of issues: they are supplementary [Hol05].

In this section, we present the main set of usability test methods listed in [Nie93].

3.3.2.1. Usability Testing

Usability Testing, or also called User Testing, is probably the oldest of the methods that evaluate with users to improve the usability of a system. Hence the similarity in characteristics with most of the other usability test methods, which are different approaches of the same idea: test with real users. Users are the best source to find out their exact problems at the time to use the evaluated interfaces [Nie93, p. 165].

The method basically consists of users performing a set of tasks with the interface, being it a prototype or a final version, while the evaluator observes their behaviour and collects information about the way the users have performed such tasks [DFAB98]. Typical data collected during these sessions are the number of errors, time needed to perform a specific task and the level of satisfaction of the users. Following, the collected data is reviewed and interpreted for a future revision of the interface design. The sessions where users try the system might be recorded to facilitate the reviewing of the evaluator, who will finally have to deliver a list of the detected problems together with specific redesign suggestions [MRT].

Usability Testing intrinsically incorporates methodological pitfalls that must be taken care of, such as reliability and validity issues. The former, is the question of whether the results would be the same if the tests were repeated, while the latter, is the question whether the result actually reflects the usability issues that were expected to be tested [Nie93, p. 165]. In order to avoid any inconvenience related to both, the reliability and validity of the results, the plan and execution of the tests requires to be carefully designed. A good preparation for a usability testing should include the following steps [MRT]:

1. **Definition of the goals**: The objectives of an evaluation can have different levels of specificity. On the one hand, goals might be generic, like for example *"the design of the product must be easy to use"* or *"the satisfaction of the end users has to improve"*. On the other hand, goals might be more specific, like for example the readability of labels, the understandability of the terminology or the effectiveness of certain features in the interface, such as navigation bars, menus, etcetera.

2. **Selection of the test users**: It is crucial that the sample of subjects that will be evaluated represents the entire final audience for the product. There are several criteria that should be born in mind, such as the expertise of the users, their age, the frequency in which they will be using the application or the experience with similar

applications. The number of participants depends on the objectives of the test. Some studies conclude that three users are enough to find out half of the most important issues of the evaluated product [MN90]. Other studies affirm that 90% of the usability problems can be found by not less than 5 test users [Vir92]. On the other hand, a third group of studies alerts of the risks of using such limited samples of users, emphasizing that the percentages of found issues for samples with 5 users can go down to a 55% in the worst cases, while the same situations turn to increase the results to 80% for 10 test user samples and 95% for 20 test user samples [Fau03].

3. **Selection of tasks and scenarios**: The tasks submitted to the users during the evaluation need to represent the activities that people will be performing in the application [MRT]. The tasks also need to be small enough to be completed within the time limits of the user test, while still being complex enough so that they don't become trivial [Nie93, p. 186].

4. **Definition of the evaluation parameters**: Before conducting the usability testing, it is necessary to define the parameters that will be used to measure and interpret the results [MRT]. These parameters are strongly related to the level of specificity defined in the first step. At this stage, different more specific methods to collect data appear, such as *"Thinking Aloud"* or *"Co-discovery"*, presented in the next section of this chapter. It is worth noting though, that such techniques do not provide reliable ways to collect data about the users' satisfaction [MRT]. For parameters of a more subjective nature, other methods such as *"Questionnaires"* or *"Interviews"* are applied (presented also in later sections of this chapter).

5. **Preparation of the material and environment**: If it is a recorded test, the proper equipment should be prepared. The roles of the experimental team members have to be established prior to the test start. Also, it is recommended to rehearse with a pilot trial to check and refine the test procedures [MRT].

In order to get the best out of the tests we must not forget that we are dealing with people and, therefore, some considerations must be taken into account. For example, it is important to remember that the purpose of the test is to evaluate the software and not the user. Another recommended procedure is to reassure that the results of the test will be kept confidential, so that the testers can feel more comfortable [Nie93, p. 188].

Advantages

- Tests with real end users [MN90].

Disadvantages

- The behaviour of the users might be affected. The method is intrusive.
- The reliability of the output is very dependant on the number of testers [Fau03].
- Requires a big amount of time.
- Finding testers eager to help is not always an easy task.

3.3.2.2. Thinking Aloud

Thinking Aloud (THA) is a usability test method where the tested users express their thoughts towards the application while executing a set of tasks. Thinking Aloud may be the single most valuable usability engineering method [Nie93, p. 195] and it has been as such for the last two decades. This fact is only explainable because of its independence from guidelines. As human behaviour evolves way slower than the technology we develop, methods that evaluate our behaviours are more keen to survive over the time [Nie12].

In a thinking aloud test, test participants are asked to continuously verbalize their thoughts while using the system, and it is with this verbalization of thoughts that evaluators are able to understand how users understand the system, making it easier to identify the end users' misconceptions [Hol05]. Thus, facilitating a direct identification of which parts of the interface are more error-prone and should be redesigned. This method aims to solve the problem of the amount of time passing between the users performing a task and expressing their thoughts towards their actions. If they express their thoughts immediately after having them, less details are lost and the role of human memory is less relevant. This is always something good considering our working memory is pretty limited [Hol05]. The success in this direction is evident when compared to other retrospective methods, which rely on the users' memory of what they had been thinking some time ago [vdHdJ03].

The influence of the method on the performance of the test participants is also significant. As an intrusive method, it distorts their behaviour but not necessarily in a negative way. For example, users might realize their own mistakes while listening to themselves, finding solutions they wouldn't have found alone [Nie93, p. 196].

People always try to appear smart and this might make them try to prepare their sentences more thoroughly than desired. A special emphasis must be put for this not to happen. The evaluator must be paying attention and prompt users to keep talking. It is essential to get the user's raw stream of thought [Nie12].

Advantages

- Independent of usability guidelines [Nie12].
- Serves as a *"window on the soul"*, letting evaluators discover what users really think on the designs [Nie12].
- No details are lost with the passage of time [Hol05].
- Not many test participants are required [Hol05].
- Preference and performance information can be collected simultaneously [Hol05].
- May help users to focus and concentrate on their task [Hol05].
- It's cheap: no special equipment is required [Nie12].
- It's robust: There is no real methodology and therefore, it is hard for something to go wrong [Nie12].

- It's flexible: Applicable at all stages of the development [Nie12].
- It's convincing: End users are *always* right [Nie12].
- Easy to learn: Basic applications of this method can be run without prior expertise [Nie12].
- Weekly user testing is completely feasible with this method [Nie12].

Disadvantages

- Hampers the recollection of information for performance measurement [Hol05].
- Different learning styles might be a burden for the users [Hol05].
- Users may feel inhibited [Hol05].
- It is a time-consuming method [Hol05].
- May give a false impression of the cause of usability problems if the words of users are taken too seriously [Nie93, p. 195].
- Causing users to focus and concentrate may distort their behaviours [Hol05].
- Unnatural situation: It is hard for the test participants to keep up talking out loud [Nie12] especially for advanced users who do not even realize how they performed certain actions [Nie93, p. 196].
- Previously thought statements must be dealt with [Nie12].
- It is an intrusive method [Nie12].
- Needs to be complemented by other usability evaluation methods [Nie12].

Constructive Interaction

Constructive Interaction or Co-discovery is a variation of the Thinking Aloud method which involves having two test users try the system together [Hol05]. This new environment has a positive impact on the users when it comes to express thoughts, since people are used to verbalizing their thoughts when trying to solve a problem together [Hol05]. Also the feeling of collaboration towards finding a solution disinhibits the test participants, who will now verbalize more less-thorough observations.

Advantages

- Much more natural situation than speaking alone [Hol05].
- Enhances the verbalization of thoughts [Hol05].

Disadvantages

- Testers may have different learning strategies [DR99].
- It is difficult to train pairs of testers to work together [DR99].

3.3.2.3. Field Observation

Field Observation is the simplest of all the usability methods [Nie93, p. 207]. It involves visiting one or more users at their workplace while trying to interfere the minimum with their work [Nie93, p. 207]. This way the observer can watch them work in their natural environment and understand how they use the product [PRS+94]. This observation must be carried out with the maximum discretion in order to be as little intrusive as possible and do not interfere with the users work [Hol05]. Ideally, the observer should be virtually invisible to ensure normal working conditions [Hol05]. In some cases, video recording might be a good option to simulate such situation, though this is not really a rule owing to the high cost in terms of time of videotape revisions. To analyze a recording might require up to 10 times more time than the actual recording which makes this time being better spent in testing other subjects [Hol05]. This method focuses on significant usability issues and therefore, one iteration per user and version of the product is normally enough. Hence the unnecessity of recordings [Hol05].

During the observation, there might be some occasions where it is necessary for the observer to interrupt the user and ask for an explanation of some sort, so that the observer can understand more easily the reason of the actions the user has made to tackle a specific task [Nie93, p. 208]. In any case, these interruptions should be avoided as much as possible. For example, a good alternative to interfering with the user's work is to write down the ununderstood action, and wait to see if it occurs again, hopefully solving the doubt [Nie93, p. 208]. It is important that the observer, who supposedly knows the system, declines to answer or solve any doubts of the users, at least, until the end of the evaluation [Nie93, p. 208].

An interesting asset of Field Observation is that during the observation, many new and unexpected ways to use the system and tackle the problems for which the tool is thought arise. Designers cannot always give for granted that their tool will unalterably be used for what they initially intended, and they can get rich feedback to adapt the system with such observations [Nie93, p. 208].

Advantages

- New applications or ways to approach a solution are discovered [Nie93].
- It is the simplest of all the methods [Nie93].
- Its application is cheap [PRS+94].
- Evaluates users while these are in their real context [PRS+94].

Disadvantages

- Despite the efforts, it is still an intrusive method [Nie93].
- Analysis after the observation, of recordings or notes, is hard and costly [Hol05].

3.3.2.4. Logs Analysis

Logs Analysis is a usability test method that involves having the computer automatically keep track of the actions of the users. Thus, being able to produce statistics and having detailed information of the use of the system [Nie93, p. 217]. This method is normally applied after the release of the system but it can also be used as a supplementary method during the development [Nie93, p. 217].

Keeping track of how the users actually use the system is tremendously useful because it shows how the users perform their work in terms of precise data which is easily and automatically collected by the computer [Nie93]. This automation offers the possibility to test a very large number of users, who might be working under different circumstances [Nie93]. The generated statistics normally represent the frequency with which users have used each feature, and also the frequency in which other significant events ocurr, such as error messages, use of rollback options, etcetera [Nie93, p. 217]. Once the most frequently used features are known, these can be optimized to improve the efficiency of the system [Nie93]. On the other hand, those features that are found rarely used or simply unused, may need to be emphasized, improved or deleted depending on their importance within the system. Statistics showing the frequency of errors during the use of the system can be used to improve the usability of future releases, tackling first and foremost those that ocurr more frequently [Nie93, p. 218]. Such statistics also emphasize the idea of improving the error messages, so that they are more understandable and constructive [Nie93, p. 218].

Logging can be achieved either by instrumenting low-level parts of the system software, such as keyboard or mouse drivers, or by modifying the software that is being logged [Nie93]. The latter approach is generally better because it allows the execution of more detailed logging, at a feature or even action level. The statistical analysis of the data can also be implemented, thus simplifying the work and providing the option to automatically alert the designers in the eventuality of severe misfunctioning of the system.

Advantages

- Can esily collect information for a large number of users [Nie93].
- Easily provides a benchmark of the usage of the various features of the system.
- Can be executed continuously and without supervision.
- Most common errors can be easily found and tackled [Nie93].
- Can analyse the data and provide a tool that alerts the designers in case of changes in the user needs [Nie93, p. 219].

Disadvantages

- Not applicable at early stages of the development [Nie93].
- Requires a big amount of testers for the data to be useful.
- It represents a potential violation of the testers' privacy.
- Requires tedious implementation work, together with maintenance.

3.3.2.5. Questionnaires

Some aspects of usability are best studied by simply asking the end users. Especially those related to the users' subjective satisfaction or possible worries, which are hard to measure objectively [Nie93, p. 209]. Questionnaires are an indirect usability test method: They do not study the actual user interface design but they collect the users' opinions about it. Indirect methods are useful for studying how users conceive systems and what features they particularly like or dislike [Nie93, p. 209].

In indirect methods, the statements users' utter should under any circumstance be considered as true or more valuable than those discoveries made with the use of alternative methods. Normally, observations on what users do are more relevant than the observations made by users where they say what they do. This has a great relation to the fact that users will always try to appear as smart as possible, and in indirect methods, users always have a reasonable amount of time to thoroughly work on their answers. But there are other causes that might bias the quality of the answers, for example, users might believe they understood concepts in a better way than what they actually did [Nie93, p. 210]. Therefore, it is important that the users answer to the questionnaires shortly after having used the system [RD83]. In conclusion, as an indirect method, the validity of the users' statements is questionable [Hol05].

Questionnaires basically consist of a set of questions presented to the users, printed on paper or digitally, whose answers are later collected. No supervisor is really required while the users fill out the questionnaires [Nie93, p.210].

Questionnaires can be distributed to the entire user audience, making this method, probably the only one to make such extensive coverage feasible. This provides a great opportunity to discover differences between various groups of user categories, and also to find out the specific needs of various small user groups [Nie93, p.211].

Advantages

- Subjective user preferences such as satisfaction can be easily identified [Hol05].
- Can be used to compile statistics [Hol05].
- No administrators are required during its execution [Nie93, p.210].
- Can be executed with an extremely large amount of the final audience [Nie93, p.211].

Disadvantages

- As an indirect method, the validity of the users' statements is questionable [Hol05].
- Requires a big number of users for the collected data to be significant [Hol05].
- Identifies fewer problems than other methods [Hol05].
- The set of questions is fixed and invariable within a test [Nie93, p. 210].

Interviews

Interviews are a variation of the Questionnaires method. In this case, an interviewer conducts an interview where the users are asked a set of questions. These interviews may be conducted over telephone or internet, but normally involve having the interviewer travel to the user's location [Nie93, p. 211]. Interviews are well suited for exploratory studies, when the goal of the tests is not very accurately defined, since the interview can be adjusted to the situation [Nie93, p. 211].

Interviews typically include general questions where users are encouraged to widely explain themselves, thus supplying a great amount of valuable quotes [Nie93]. It is important for the success of the interview that interviewers stay neutral, not agreeing or disagreeing with users' statements. Also, the interviewer should not justify the behaviour of the system. The questions should be queried in a way that encourages users to reply with full sentences rather than with a boolean "yes" or "no" [Nie93, p. 211].

During the interviews it might be interesting to ask users to recall critical incidents they had while using the system. This kind of incidents normally happen when some of the features of the system are particularly poor, and therefore, it is a good opportunity to get better feedback on how to tackle the element that caused the issue [Nie93, p. 212]. During the course of the interview, the interviewer can continously evaluate the user's answers, and rephrase questions that may have been misunderstood. The duration of these interviews might be variable, especially because the interviewer can sense easily if the user has had enough of it, something that cannot be controlled in questionnaires [Nie93].

The same way it happens with the questionnaires, the users' answers should not be considered necessarily as true. Users tend to reply what they think they should reply and not what they really think, especially in sensitive questions where certain answers may be embarrassing [Nie93]. Also, the fact that users give for granted that the interviewer, in some way, forms part of the development team, may inhibit them from criticizing the design.

Interviews can be conducted at any time during the development of the product, though it might be more suitable at final stages. This is because users are expected to have tested a prototype before the interview starts, and therefore, at least an early version should be available [YS05].

Ideally, in any interview scenario, the test participants should be part of the actual user population who is going to be using the product. Thus, making the output to be more revealing that if this were not the case. In the case that this is not possible, potential designers should consider interviewing other colleagues [YS05].

For non-expert interviewers, a semistructured set of questions is recommended. A useful structure for usability interviews could be based on, for example, the checklist proposed by Ravden and Johnson [RJ89]. Using the sections of such checklists as guidance, one can keep flexibility while ensuring thoroughness in covering all aspects of usability [YS05]. It is important that interviews are fluid and not seen as a list of questions that need to be asked. Prepared guides of sets of questions should basically be seen as an agenda to ensure that all the aspects are covered.

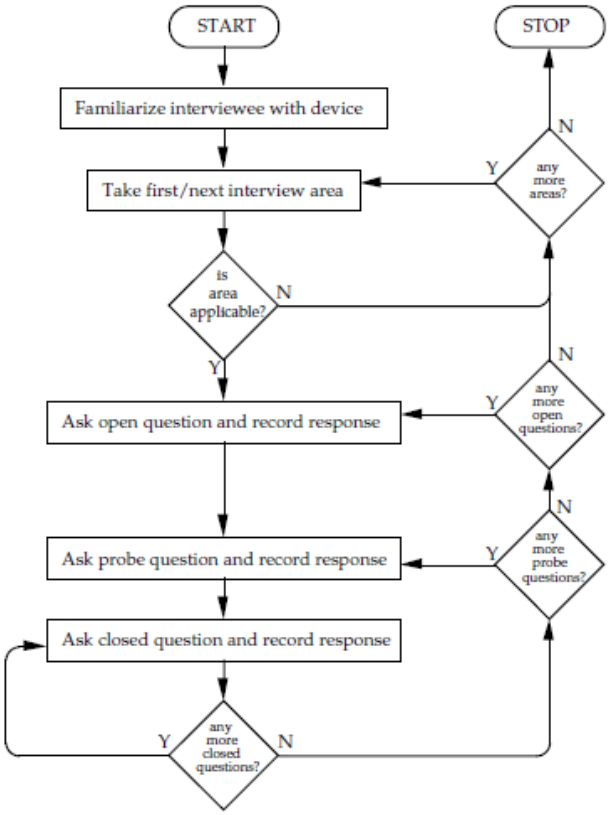


Figure 3.4.: *Flowchart of the usability interview process [YS05].*

It is recommended to mix different types of questions, from very open queries (such as *"What do you think of this feature?"*) through probing questions (such as *"Why do you think so?"*) to more specific ones (such as *"What do you think this button is for?"*) [YS05]. It may be useful to proceed in a protocol similar to the one proposed in Figure 3.4. The main idea is that the interviewer opens a line of inquiry with an open question and then continues with it, through more specific questions, depending on how alive the thread is. When the line of inquiry is exhausted, the interviewer must dynamize the interview opening a new line. If the interviewer can manage to meticulously go through every aspect of the device, then the output will be more complete [YS05]. Nonetheless, the evolution of the interview depends on too many variables to be considered in advance, and an incomplete interview does not necessarily mean it was a bad interview. The level of detail achieved by the conversation might provide better feedback than other interviews that have gone through the whole system. This especially applies to systems with a wide variety of features. Also, dynamic interviews, where interviewees might show special interest in certain features, highlight the trade-off between specificity and completeness of the interviews. Finally, it might also be interesting to have a prepared data sheet to fill in the responses during the interview. Interviews can also be recorded, though going through hours of conversations normally is in a tedious task.

The expertise of the interviewers in both, interviewing processes and the product they are evaluating, has an important impact in the success of the interviews. This is easiliy compensated by the ease of learning to use such a method. It has been proven that combined training and practice times for this method were relatively lower than those of other more complex methods [SY99].

Advantages

- Is more flexible than Questionnaires: two-way communication [Nie93, p. 210].
- Is more adaptable than Questionnaires: Interviewers may add/remove additional questions when the situation is favourable [Nie93, p. 210].
- Helps to deeply understand users.
- Generates immediate results [Nie93, p. 210].
- Applicable at almost all stages of the development [YS05].
- People are familiar with how interviews work [YS05].

Disadvantages

- Requires more usability staff time than Questionnaires [Nie93, p. 210].
- The compilation of data can be difficult if the interviews are too variable between one another [Nie93, p. 210].
- It is time consuming and expensive.
- Can yield different answers from the same participant.
- The success of its application highly depends on the expertise of the interviewer.
- It is difficult to arrange suitable time and users commitment.
- As an indirect method, the validity of the users' statements is questionable [Hol05].
- Requires, at least, an early version of the product [YS05].

3.3.2.6. Focus Groups

Focus Groups are a usability test method that can be used to assess user needs and feelings towards a system. This method consists of carefully planned discussions, designed to obtain the perceptions of the group members on a defined area of interest [KLB04]. Together, the users will discuss new concepts and identify issues of the system. The groups should be formed with about 6 to 9 persons, who are carefully selected depending on their individual characteristics, and a moderator, who guides and facilitates the discussion following a predefined structure [Nie93, p. 214]. Focus groups enable the participants to build on responses and ideas of others, increasing the richness of the information [LM03] and allowing the appearance of users' spontaneous reactions and ideas through interaction [Nie93, p.214].

During Focus Groups sessions, qualitative information about the objects of study is principally produced [KLB04].

The main steps of the Focus Groups process should be [KLB04]:

1. **Definition of the research problem**: The focus group method has been proven to be more suitable for obtaining feedback on new concepts, developing questionnaires, brainstorming, collecting and categorizing potential problems, discussing how models are presented or documented, and to discover special user interests or worries. The method is not suitable for hypotheses' testing, taking decisions, obtaining quantitative data and studying complex issues that are difficult to grasp in a single session, because all the participants need to feel comfortable within the group.

2. **Plan of the focus group session**: The sessions for this method normally last between two and three hours and it is recommended to create a predefined structured schedule. The number of issues to be covered should be limited so that there is enough time to discuss each of the chosen issues. This way, participants have sufficient time to comprehend the issue increasing the quality of the following discussion.

3. **Selection of the participants**: The experience in both, the matter and the participation in focus groups, and the insights of the participants have a great impact on the success of the final output. Therefore, it is important to recruit a representative, insightful and motivated group of participants. The users should normally represent part of the final audience for the product, though this does not necessarily need to be the rule.

4. **Strategy for the focus group session**: The focus group session should be conducted so that there is enough time to discuss all the planned issues, but also making sure that everyone has time to express their opinions. The session should be started with an introduction where the goals and ground rules of the session are presented to the participants. The discussion can take many different forms, such as structured discussions, where the role of the moderator is more important, brainstorming techniques, voting on preferences, comparison games or even role plays. In [LM03] up to 38 different approaches to supplement traditional focus group techniques are presented. The sessions can be recorded in a wide variety of ways, being the most com-

mon video and audio recording or additional observers taking notes. It is also recommended to arrange a debriefing session immediately after the session so that fresh observations are not lost with the passage of time. Moderators taking notes during the session is a practice that should be avoided when possible, as it may interrupt and hamper the dynamic of the discussion.

5. **Analysis and report of the data**: Normally, only qualitative data is produced out of this sessions. This data should then be analyzed consequently, with any of the methods for qualitative data analysis.

This method is currently widely used in many different fields such as market research, product planning, political campaigning, business services defining or system usability evaluation [KLB04]. The abundance of literature makes it easy adoptable and consistently used.

Advantages

- It can be applied at all stages of the development [Nie93, p. 214].
- Its application is cheap [KLB04].
- Its quickly performed [KLB04].
- Discovers new insights [KLB04].
- Aids recall: Statements made by participants might be confirmed by other participants who alone would not have reminded such observation [KLB04].
- It is cost-efficient [KLB04].
- The reasons of why participants think the way they do are discussed more deeply [KLB04].
- Encourages the cooperation between participants [KLB04].
- The output of the discussions is normally very easy to analyse and understand [SSR06].

Disadvantages

- Group dynamics might bias the output [KLB04].
- Groups tend to be small, and therefore, it is difficult to generalize the results [KLB04].
- Social acceptability might bias the quality of the expressions of the participants [KLB04].
- As any other discussion, complex points are not always understood by all the participants [KLB04].
- Can be misleading: That a majority of the participants in the session agree on something does not necessarily mean that is true.

3.3.2.7. User Feedback

For already working systems, the users themselves can be an important source of usability information [Nie93, p. 221]. User Feedback basically consists of developing a platform for users to write about the issues they have found in the system, their complaints, observations for improvements, desired changes, etcetera. It is important to note that the group of users is never chosen by the analysts in this method, as sending feedback is voluntarily done by the users. Normally, one of the main motivations for users to provide feedback is to relieve their frustrations while using the system, so many of the incoming observations might literally turn into complaints. Analysts should not be daunted by such perspectives, as those observations are still useful at the time to provide information on problematic features of the interface. On the other hand, in the case where possible improvements or changes are suggested, these should be deeply considered, as many users tend to desire systems to be customized to their individual needs, which may turn out in an even worse solution.

Systems that provide a window for feedback to be delivered, can also make snapshots of the users' system situation, as complaints are normally written after the problem has happened, or while the issue is not letting the users to proceed with their tasks. It is important that the snapshot provides as much information as possible because users rarely provide technical descriptions of their situation [Nie93, p. 222].

Customer support help lines can also serve to collect feedback from the users together with statistics about frequent user problems. In the last years, the releasing of beta versions has become very popular. These beta versions are prototype versions, which might run parallely to previous final versions of the product, and can be tested by a smaller group of so called *"beta-testers"*, whose purpose is to provide feedback on the product. This methodology is useful because the feedback normally arrives in time to improve the upcoming full release of the product. On the other hand, it has to be taken into consideration that this feedback will still arrive too late to do as much good as other usability engineering methods would have done in earlier stages. Thus, it is important of complementing this method with other methods that can be applied at the beginning of the development [Nie93, p. 223].

Advantages

- Shows users' concerns [Nie93, p. 221].
- Feedback is received without any special efforts to collect it [Nie93, p. 221].
- Quickly shows changes in the users' needs [Nie93, p. 222].

Disadvantages

- The received feedback is not representative of all the users of the system [Nie93].
- May give a false impression of the cause of usability problems if the words of users are taken too seriously [Nie93, p. 222].
- Requires to be complemented by other usability evaluation methods [Nie93, p. 222].

Part II.

TACKO

4. TACKO

As we have seen in Chapter 2, one of the main challenges of modern informatics is the effective management of the large amount of digital information being produced nowadays. This means that new alternative knowledge organization systems for easily finding and retrieving stored data, where traditional organization systems do not apply anymore, are required. Concretely, the advent of the Web 2.0 and other collaborative technologies, has shown the obsolescence of conventional systems, such as the popular hierarchical organization systems, which have proven to be inflexible at the time to represent and structure the different information needs and perspectives on a collection of documents [MNS12a].

Due to this necessity, and with the help of nowadays improved technical possibilities, new approaches such as faceted classification systems or folksonomies generated by social tagging systems, have been found more suitable in recent years. Many tagging systems tackling the difficulties inherent in tags from multiple different perspectives are currently being proposed and developed. One of these, is the TACKO project[1].

TACKO[2] is a multi-faceted context-dependant knowledge organization system based on the flexibility tags provide, yet incorporating elements of hierarchical and faceted organization schemes, allowing the application of different independent categorization dimensions to collections of information resources [MNS12a].

In this chapter we will provide an overall summary of the main motivations and challenges of this approach, for a better understanding of the usability evaluation we conducted on its interface. In *Motivation* an overview of the main elements incorporated in the system and how it can benefit from their strengths is discussed. In *Model* the main ideas of the system are summarized and illustrated with an example. Finally, we present its current *Interface* in detail, presenting also its latest major changes. We leave its justification and other observations for the Chapter 6, where its usability is evaluated.

[1] `http://wwwmatthes.in.tum.de/wikis/sebis/tacko` (Last accessed on 18/08/2012)
[2] Acronym for *TAg-based Context dependant Knowledge Organization*

4.1. Motivation

Below, we briefly present the main strengths and weaknesses of some traditional organization schemes, some of whose characteristics are applied, enhanced and combined within TACKO. Additionally, we provide a concise explanation of the concept of *implicit relations* and its three kinds, one of which is currently being applied in TACKO. Finally, we state the main requirements for a system combining the forementioned characteristics: TACKO.

Hierarchies

In hierarchies, individual information resources are associated to a set of categories, each of which has exactly one parent category and may contain none or several subcategories. The resources are naturally assigned to one unique category, though symbolic links can alter such structure. Hierarchies can easily be represented as tree structures, reflecting the organization of physical items in space, and thus, are easily understood by the users. The existence of a unique unambigous path from the root to each resource helps the users to easily find the location of such resources within the hierarchy. On the other hand, it is cognitively challenging for the users to decide in which single category a resource fits, both at the time to store and retrieve such resources, because these can normally apply to more than one concept. This problem is emphasized in collaborative systems, where users' different opinions and perspectives might influence the structure. Moreover, users are possibly required to browse to maximum specificity when different levels of categorization are defined. The order of these levels of categorization is imposed making the management and the reorganization of a hierarchy complicated and efortful [MNS12a].

Tags

Tags are simple text labels that can be arbitrarily assigned to individual resources. This entails a great flexibility and is especially useful for scenarios where large quantities of resources are organized by several independent users. The structure result of this practice is known as a *folksonomy* (See Section 2.1.1.4). With tags, resources can be assigned to several possibly overlapping facets. Nonetheless, due to their unconstrained nature, several semantic and cognitive considerations need to be taken into account (See Section 2.1.2). Furthermore, tag navigation options are very limited. The lack of explicitly established relations between tags, leaves tag co-occurrence frequency as the only means in which to base tag-relatedness. An extensive discussion on this topic can be found in Section 2.3. Finally, an important aspect of systems using tags is whether they represent a *broad folksonomy*, where users manage their own tags for each resource, or a *narrow folksonomy*, where the set of tags of each resource is shared by all the users. Since TACKO intends to support collaboratively managed collections, the latter is assumed [MNS12a].

Facets

We already broadly presented the concept of facets in Chapter 2.1.1.3, notwithstanding we find appropriate to analyze their characteristics within the context of TACKO. Facets are independent orthogonal dimensions of classification, represented by sets of concepts, that can be hierarchically organized within a facet, to which individual information resources

are attached. Facets allow dynamic filtering of the resource collections using arbitrary combinations of concepts from different sets. Their flexibility at the time to locate resources, such as goods and services, among large collections has made them very popular on modern e-commerce sites. However, their application in systems where users can adapt them to their needs is yet unknown, and this is one of the goals of TACKO [MNS12a].

4.1.1. Implicit Relations

Implicit tag relations is the notion of those relations between tags that can be induced from the users tagging practices. Namely, users can consciously alter the state of the system, i.e, the relations between the tags, without explicitly defining their relations but with the sole application of maintenance controls that implicitly reflect such relations.

These can be really useful to improve the accessibility of contents in tagging systems without significantly reducing the flexibility of tags, because their use improves the findability of resources through direct searches and more structured visual representations, such as tagclouds, can be generated [MNS12b].

Considering that navigation in a tagging system can be understood as a series of steps where tags are added or removed from a filter, it is important to note that these implicit relations only apply to certain *contexts*, defined by such filters, of the *folksonomy* [MNS12b].

In [MNS12b] three different kinds of implicit tag relations are formally introduced. Nonetheless, we will here summarize such elaborations:

- **Subsumption**: A tag subsumes another one when the resources assigned to the second one represent a subset of the resources assigned to the first one. This relation is useful to map *hypernyms*, i.e superordinates, or *holonyms*, terms that denote a whole whose part is denoted by other terms, among others. In *folksonomies*, where both specific and general tags are used, it is appropriate that all the resources assigned with the specific tag also contain the respective general tag. It is also important to note that a tag can be subsumed by several other tags that are not in a subsumption relation between one another. It is recommended to hide subsumed tags because redundant information is trimmed and the room they would be using in the interface can be reused by other tags that would have possibly been ignored otherwise.
- **Equivalence**: Two tags are equivalent in a certain context defined by a filter if both or neither of them are assigned to every single resource at the same time for such context. This relation is useful to map *synonyms*, terms that are semantically equivalent.
- **Mutual Exclusion**: Two tags are mutually exclusive when the intersection of their respective sets of assigned resources is empty. This relation is useful to narrow down its resulting set when navigating a *folksonomy*, because the tags in such sets typically belong to the same facet. It also helps to disambiguate homonyms.

Advantages

Following we summarize the main benefits of implicit tag relations [MNS12b]:

- **Increased recall for searches**: Any tag used in the system can be easily assigned to all the resources it applies to, improving the search features.
- **Improved navigation options**: Search result sets can be refined in multiple ways: tags being subsumed by more general tags don't need to be shown, equivalent tags can be grouped or represented by an arbitrarily chosen tag of such group and mutually exclusive tags ease the identification of facets. Such refinements minimize the cognitive efforts users have to put at the time to scan the final shown tag sets. Also, the reduction of tags that express similar or related concepts makes room for less frequent tags, enabling new concepts to come on scene, thus broadening the pool of concepts and allowing the user to navigate to resources that would be unreachable otherwise. Other considerations such as the concealment of tags being assigned to all the resources of the current context can be applied.
- **Emergent organization structures**: Users can harmonize the usage of tags in manageable subsets of resources, typically their own sets of resources. Also, implicit relations enable the apparition of complex structures that could hardly be expressed using classical knowledge representations. Finally, they flexibly adapt when new resources are added to the system.
- **Portability and universal applicability**: The fact that such relations are not explicitly stored makes it conceivable that they can be applied in all systems. However, though this is always true for their representation, these systems should comply a set of requirements that ensure the possibility to manage and use such relations. This set of requirements is introduced in the following section.

Requirements

Every tagging system that aspires to benefit from implicit tag relations should comply with the following two requirements [MNS12b]:

- **Advanced user interface**: The user interface of such systems should provide advanced support for browsing tagged contents, for adding new resources to the collection and for analyzing, refining and establishing tag relations. Moreover, advanced tag presentation techniques should be updated after each navigation step, where the suggestions adapt to the most frequent tags in the current context, incrementally refining the categorization. Additionally, the system has to assert that existing tag relations are preserved for every alteration in the state of the system. It is also recommended that the interface provides the possibility to narrow the filter in such a way certain tags are excluded.
- **Efficient recognition and creation of tag relations**: Tag relations should be dynamically detectable in arbitrary contexts. The massive assignment and removal should be efficiently supported also for large sets of resources. This can be done using in-memory databases or transitionally storing operations in the form of rules that are

applied to the collections when possible and transparently to the users. TACKO is currently implementing the latter approach.

Challenges and limitations

This approach also involves some challenges and limitations [MNS12b]:

- **Limited expressivity**: Distinction between part-whole and subtype relations cannot be done because both are expressed as *subsumptions*. Furthermore, the representation of such relations in more explicit knowledge representations is very limited.
- **Unintended tag relations**: A distinction between meaningless co-occurrences of tags and relations that have been consciously established needs to be done. Defining a threshold below which implicit tag relations are considered *false positives* and ignored by the system solves this limitation to a certain extent, though finding such thresholds is not trivial.
- **Scalability and applicability**: To scale such approach to thousands of millions of resources is technically challenging. Also, social dynamics need to be treated, e.g adding regulations on how can tags of strangers' resources be modified and how the possible disagreements are handled, possibly with restricted access rights. The massive expected increase in the total number of tag assignments, e.g synonyms are stored redundantly for each resource leading to a multiplication of the average tags per resource, also presents a technical challenge.
- **Usability**: Designing and implementing an interface that allows the exploitation of the full flexibility of implicit relations while providing an easily understood, learnt and used way of managing and navigating through *folksonomies*, is not trivial. Such innovative systems generally require innovative interfaces users not always initially favour.

4.1.2. Requirements

The aim of TACKO is to create a new organization scheme that combines hierarchies, tags and facets, benefiting from their respective structural advantages, and that is easily adaptable by users. In other words, a system that merges the flexibility of tags with the organizational assets from faceted classification and the navigability options hierarchies provide, where users are capable to locally apply hierarchical relations between categories, that is to say, subsets of resources can be reorganized without affecting the whole collection. Additionally, for the application of such concepts, TACKO employs the forementioned notion of *subsumption* to ease the maintainance of the tag connections consistency within the scheme. Obviously, the resultant scheme should be easily learned and applied by the users [MNS12a].

4.2. Model

Subsequently, we summarize the main details of the TACKO model and provide an example to illustrate them. For a more formal definition, see [MNS12a].

- **Tags as basic means for categorization**: Tags are the primary means for categorization in TACKO. Others techniques, such as hierarchies and facets, are defined afterwards over these tags. In TACKO, there is no limitation in the number of tags that can be defined for every resource and it is not relevant who assigned them.

- **Accessing resources**: The collection of resources is always accessed with the use of filters. Filtering the collection is a restriction some resources will not comply, narrowing therefore, the presented set of resources. A filter is a set of tags, and thus, we say a resource matches a filter if all the tags in the filter are also present in the set of tags assigned to the specific resource. The set of matching resources is known as *result set*. Clearly, the more tags are added to a filter, the narrower the result set is. In this case, we would say a filter becomes more specific. On the other hand, removing tags from the filter, results in wider sets of resources, and so, we say such filter becomes more general. The notion of *complementary tags* is introduced to allow users to view and modify resources for which a particular tags is *not* assigned. Such tags can be understood as the negation of the actual tags. Filters may contain both tags and complementary tags at the same time. Complementary tags also impose a restriction on the result set, and therefore, this is also narrowed when those are added to the filter. Finally, it is important to note that the possibly defined tag relations only apply in the context of a certain filter and its specializations.

- **Hierarchical tag relations**: Often, clearly hierarchical relations between specific tags, such as *part-of* and *is-a* relationships, appear. To be able to consider these, TACKO allows *subsumption* relations (See Section 4.1.1) also applying in the context of certain filters. Applying such relations enables the definition of different levels of categorization, such as facets, independent of the hierarchies. This way, single resources can be assigned to multiple categories, that can at the same time be subsumed by several other categories. Subsumption relations do not apply to filters containing complementary tags, because the complexity of the resulting organization structure would notably increase. Users would have serious difficulties to understand it and from a computational point of view it would be difficult to implement in an efficient way.

- **Facets**: TACKO allows to explicitly define tags belonging to the same facet in the context of certain filters. Facets containing only one tag are also allowed, because they still add the simple dimension of categorization of whether a tag is assigned or not. As opposed to subsumption relations, facets do not restrict the way tags are assigned to resources.

4.2.1. Illustration

Following we illustrate some of the main newly introduced concepts of TACKO in the context of the example shown in Figure 4.1. For a more elaborate and detailed illustration see [MNS12a].

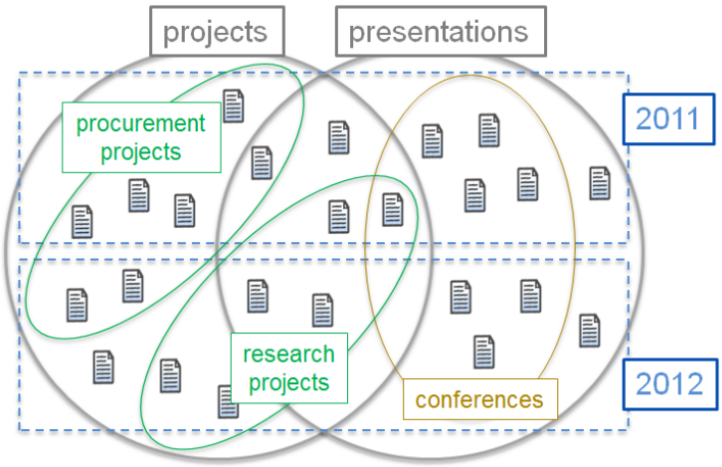


Figure 4.1.: *Schematic illustration of context-dependant tag-relations [MNS12a].*

In Figure 4.1 a tag is assigned to a certain resource if such resource is within the shape representing such tag. Additionally, similarly formed and oriented shapes represent tags belonging to the same facet.

At first sight we can easily identify two facets of two tags: one formed by *projects* and *presentations*, and another one formed by *2011* and *2012*. Facets can be *single-valued*, which means that there is no resource assigned with more than one of the tags of the facet, that is to say tags are exclusive within the facet, i.e the facet formed by the years, or *multi-valued*, which means that resources can contain more than one of the tags of the facet, i.e the facet containing *projects* and *presentations*. Moreover, the tags *research projects* and *procurement projects* are subsumed by *projects*, i.e their respective shapes are within the shape representing *projects*, ensuring better future searches. Also, these two tags form a facet within the context of the filter *projects*.

Facets have no name because their meaning is implicit in their tags and context. Defining facets in the context of certain filters enables tags to belong to different facets in different contexts. For example, *vegetarian* could be in a facet together with *vegan* and *meat* for the context of *recipes*, together with *chinese* and *indian* in the context of *restaurants* and together with *smoker* or *handicapped* in the context of *persons*. The representation of such categorization is possible in traditional hierarchies, though limited. In TACKO not only it is easy to apply but it would be also possible to define a facet consisting of the tags *recipes*, *restaurants* and *persons*, in the context of *vegetarian* [MNS12a].

4.3. Interface

TACKO is a system intended to be used collaboratively and for large collections of resources that may be of different nature. For example, in the prototype version presented in [MNS12a] it is applied to blog posts, wiki pages and shared files, and therefore the interface has to provide extra tools to support these resources, their management and also their visualization. However, for the usability evaluation conducted in this work a simplified prototype is proposed, with the goal to emphasize the tools regarding the maintenance of the resources' tags in the context of TACKO. Concretely, the interface presented below enables the management of the tags of sets of collections of bookmarks imported from Delicious.com[3] and images imported from Flickr.com[4]. Only the tags of these resources, and their relations can be managed, but options for resource management should be obviously added in future versions. Also, the necessary tools for the use of the *subsumption* relations are incorporated.

Also, due to the constant development and application of improvements on the different features of the interface, we find necessary to baptize the different main upgrades the interface has undergone in the timeframe of this work for the sake of the interface usability evaluation explained in chapters 5 and 6. We will first present the overall current version, which we will refer to as *version 4*, in detail. Afterwards, we will present the main previous 3 versions, and their differences in relation to their respective previous versions, in the manner of a *changelog*. We necessitate to make such distinctions because their usability might vary and they have been separatedly tested. This will help us get a better idea of how the interface has been evolving as well as to see how some of the evaluation findings of this work have been already taken into consideration. It is important to note that the defined *versions* only apply in the context of this work and that they are considered as other simple upgrades in the overall project of TACKO. Furthermore, previous and more generally applied *versions* of TACKO can be found on [MNS12a] and [MNS12b], the latter being the oldest documented version.

Finally, it is important to note that we concentrate on the explanation of the interface features and their use, leaving its justification and other possible observations regarding its usability for the chapter 6.

[3] http://www.delicious.com (Last accessed on 18/08/2012)
[4] http://www.flickr.com (Last accessed on 18/08/2012)

4.3.1. Version 4

Figure 4.2 shows a screenshot of the version 4 of the interface, which includes all the necessary tools and controls to browse the set of contents, in this case a collection of pictures imported from Flickr.

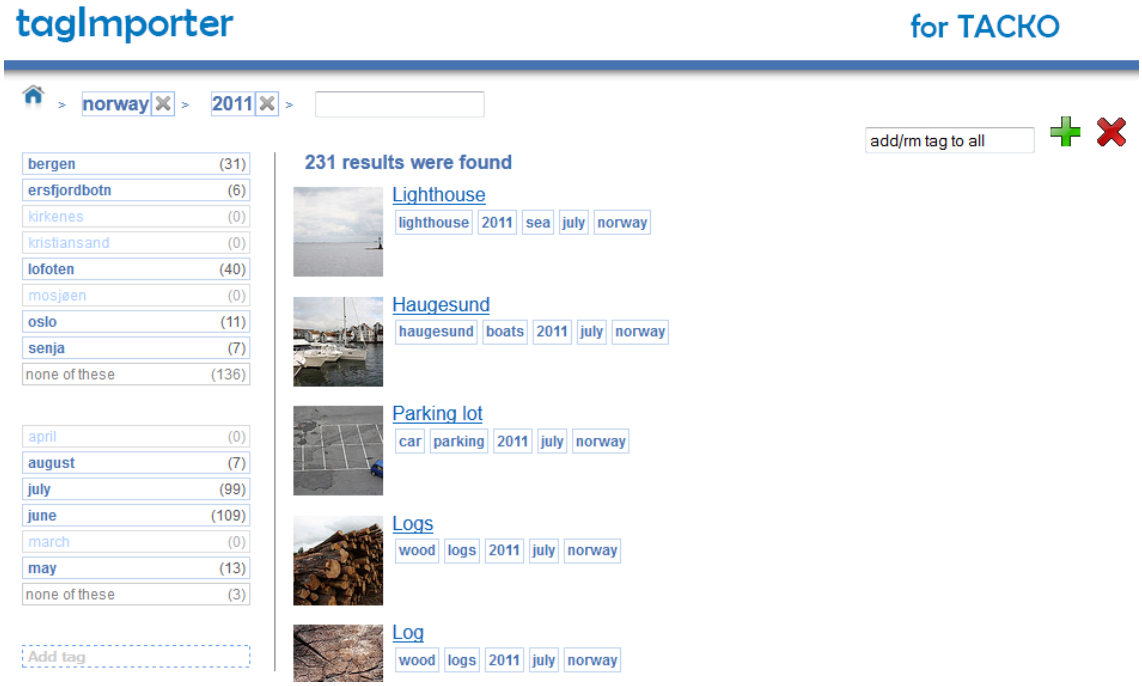


Figure 4.2.: *Screenshot of the main overview of the version 4, showing the faceted navigation and result set for the context of the filter defined by "norway" and "2011".*

As we can see, immediately after the header comes the breadcrumb. This breadcrumb contains the tags that form the filter for the current context. Breadcrumbs typically define *paths* that amalgamate the different relevant navigation steps. In this case, the different tags that have been chosen to filter the set of results. Although the order of these tags does not affect the resulting set of matching resources, it is intended to reflect the order of the navigation steps. At the end of the breadcrumb, an input text field is provided so that new tags can be added to the filter. For this purpose, and to assist the users, autocomplete suggestions are provided, for both, speeding up the process and also to alleviate possible spelling differences. This autocomplete dynamically updates depending on the input introduced by the users and the possible matches found in the overall set of tags of the system. This feature is especially useful for quickly refining the filter in situations where the users already know which tags to apply. Tags can easily be removed from the filter by clicking on the X next to them. By clicking on one of the tags in the breadcrumb all the tags to the right of such tag are removed from the filter, easily permitting to remove a certain last set of tags. This generally allows users to easily go back to previously visited environments or contexts and allows to recover from false steps. For example, in Figure 4.2 clicking on *norway* would immediately remove *2011*, thus modifying the context filter.

Similarly, clicking in the *home* icon would delete all the tags of the filter and directly go to the empty filter context, from now on denominated *Home*.

Immediately below this breadcrumb at the top right part of the Figure 4.2 a small text input and two icons, a green "+" and a red "x" can be found. As the default text states, it is a tool to add or remove a tag from all the resources of the list. A tag is introduced in the input and then, by clicking the "+" it gets added to all the resources of that context, or by clicking "x" it is removed from all the resources containing it in such context. Such input also incorporates an autocomplete function that provides suggestions. This is a massive operation and depending on the amount of resources of the context it might take several seconds or minutes if no especial technique is applied[5].

The structure on the left side of the screenshot, corresponds to the representation of the faceted navigation options that apply to the current context filter. In the example, two facets are shown. The first can be considered to represent the concept of *places in Norway*, because it includes cities, towns and islands from such country, while the second clearly represents the concept of *months*. The former is explicitly defined for the context filter of *norway*, while the latter is defined in the empty context filter. The number next to each tag represents the amount of matching resources for such tag in the current context. If a tag has no matching resources it is depreciated with a less eye-catching light blue. It is important to note that the different tags of a facet do not necessarily need to be disjoint. For example, in the first facet it could be possible to have tags representing towns situated in the Lofoten Islands, and thus, having two tags matching the same resource. By clicking a tag of the facet, this will immediately get added to the context filter and the page will be reloaded with the new context, that is, with the facets of the new context and with the new matching set of results. To determine which facets are shown in each context, more general context filters are considered. If any of these has a defined facet, this is shown in the current more specific context unless one of its tags is already part of the filter, because it probably cannot be used to further refine the filter. Concretely, this is only true for facets with disjoint tags. For facets with overlapping tags, further refinement would still be possible, but for this version it has been considered negligible. Additionally, tags that are subsumed by others are hidden.

At the end of such structures, an input field stating "add tag" is provided to manually add tags to the facet. An additional lonely "add tag" appears at the end of all the facets, and enables the creation of a new facet by simply adding its first tag. For every "add tag" input, autocomplete also makes his appearance. Moreover, a set of the up to 5 more possible appropriate tags is suggested. For the cases when this is not enough, a "more suggestions" option will expand the list to up to 20 possibilities. Tags can be removed from a facet by clicking on the "x" that appears on the right when hovering them. Both adding and removing a tag from a facet will affect the facet in all the dimensions containing it. For example, if we remove the tag *august* from its facet in Figure 4.2, and then we remove the tag *2011* from the filter, we will still find the month facet without august. Narrowing the filter behaves similarly. To delete an entire facet, its tags have to be removed one by one.

[5] In the latest versions a new system that transitionally stores operations in the form of rules that are applied to the collections when possible and transparently to the users, has been implemented, improving the efficiency of such tool.

Figure 4.3.: *Example of the use of a "none of these" text label, showing the altered faceted navigation, breadcrumb and result set for the context of the filter defined by "norway", "2011" and several negated tags.*

Every facet incorporates a *none of these* grey text label after the set of tags. Clicking this option will add a restriction to the filter meaning that non of the tags of the facet whose text label was clicked should be shown. In other words, all the resources that haven't been categorized with any of the tags of such facet are displayed, thus facilitating the categorization of such set. In Figure 4.3 an example of this is shown. None of the three listed images happens to have any of the months of the negated facet. This tool is not only useful to assign new tags to resources that are missing it, but also to find concepts already existing in a set of resources that are missing in a facet. Nonetheless, these are not the only applications of such tool. Simple visualization of resources not containing a certain tag has been found very useful as well. As we can see in Figure 4.3, the negated facet remains but their tags are shown in red. The other facets applying to the same filter, neglecting the negated tags, also appear. In this manner, the users can still see the negated tags to easily identify which ones may be missing in the set of resources. Adding such tags to the resources, will make them not comply with the restriction anymore, and therefore, they disappear from the result set. This way, the list of unrelated resources to concepts of a certain facet is minimized. Obviously, adding tags to negated facets that were found in the set of resources will also make the latter disappear from the list. It is important to note that removing a tag from the list of negated tags does not only remove them from the restriction, but also from the facet where they were found. In other words, removing *august* from the negated list of tags in Figure 4.3 will not only make resources tagged as august appear, but also will mean that august is not a concept of the facet *months* anymore.

Finally, the body of the page contains the matching result set of resources for the current filter in the form of a list, divided in pages of 10 resources each. At the bottom, a pagination allows to navigate through these pages. At the top, a counter shows the number of matching resources with the current filter. Every resource is represented as follows: The title of

Figure 4.4.: *Representation of a single resource, when being hovered, showing its preview, title and associated tags.*

the resource, that when clicked, links to the page of such resource in Delicious.com in the case of bookmarks, and Flickr.com in the case of images. Immediately below, the set of attached tags is shown. For resources of the type image, also a preview version of the photo is shown at the left. This image can be seen bigger if clicking on it, and minimized again by a further click. In Figure 4.4 an example of a flickr photo can be seen when hovering over the resource. By clicking anywhere on the highlighted bar, an edit mode is accessed. This edit mode basically enables the management of the tags of the resource. Already existing tags can be deleted by clicking on them. Also, an input implementing autocompletion enables users to manually add new tags. If such tags are part of a *subsumption* relation, their related tags are automatically added.

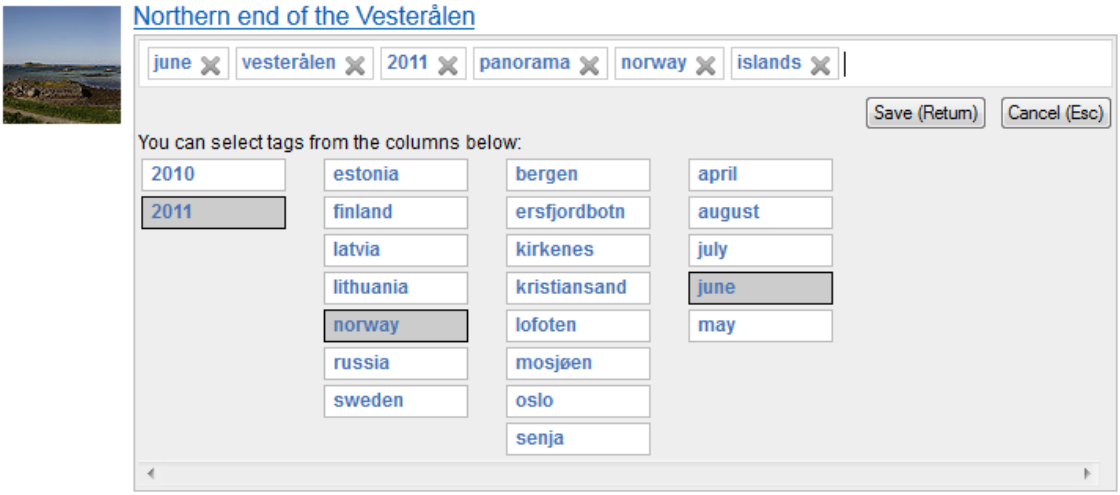


Figure 4.5.: *Edit mode for a certain resource. The faceted suggestions can be seen below.*

We can see the edit mode for a resource in Figure 4.5. In the edit mode, a set of suggested tags is presented in the structure of facets. These suggestions are a quite advanced means to obtain possibly related tags. Tags can be simply added to and removed from the resource by selecting or deselecting them from such structures. Although how these suggestions are created is not a matter of this work, it is important to know what they represent. First, the set of facets of the current context is shown highlighting those tags that are already assigned to the resource. Additionally, new alternative facets appear. These facets are those that are defined in more specific context filters formed by part or the totality of the tags assigned to the resource being edited. In other words, it provides a manner to see more specifically defined facets without needing to navigate to the respective contexts.

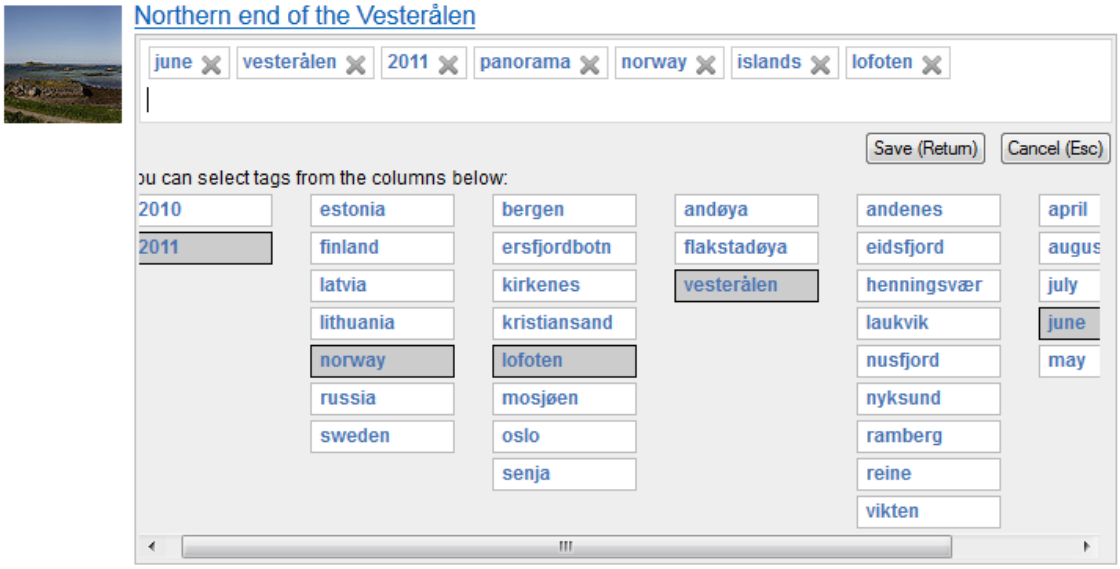


Figure 4.6.: *Edit mode for a certain resource. The faceted suggestions can be seen below and enhanced in relation to Figure 4.5.*

To exemplify this tool, in Figure 4.6 we have added the tag *lofoten* to the same resource shown in Figure 4.5, to see how its faceted suggestions are affected. Subsequently, we can observe that two new columns of tags have appeared. The column starting with the tag *andøya* represents a set of individual islands all of which are known under the name of the Lofoten. On the other hand, the column starting with the tag *andenes* represents a set of towns located in the Lofoten Islands. Basically, what has happened here is that after adding the tag *lofoten* to the resource, the facets that were defined in the context filter for this tag, or any of the combinations of the tag with other tags of the resource, are now displayed. Curiously, we can observe that the exemplified resource already contained the tag *vesterålen* categorized within the first of the new facets.

All tags in both, the structures of the left side and the resources list, are draggable. Dragging can be generally understood as "copy into" except for when a tag is dragged from one facet into another where it is a "move". Additionally, tags can be dragged into other tags within the faceted structures of the left side, representing a "move into" which basically moves the tag from wherever it was, into the more specific context defined by the current context filter extended with the tag where it was dropped into. In other words, if immediately after dragging a tag into another you click on the second one moving to the more specific context, you will see that the dragged tag now appears as a new facet. This enables the rapid creation of new facets in more specific contexts. Besides these two exceptions, the act of dragging a tag will mean a copy of that tag is created in the target resource or facet. Basically, tags can be dragged from facets into resources, otherwise, among resources and among facets. Dropping a tag into the lone "Add tag" at the end of the menu of facets will mean the creation of a new facet.

Figure 4.7.: *While dragging a tag, in this case "parking", facets display some "drop here" indicators.*

In Figure 4.7 we can observe the indicators that appear in the facets to indicate where should a tag be dropped in order to add it to them. These only appear while a tag is being dragged. In the example, *parking* is being dragged from a resource.

Additionally, dropping a tag into another within a facet introduces a new *subsumption* relation. This recalls the way drag and drop options are handled on folders in traditional file explorers. For example, if we drop the tag *bergen* into *norway*, *bergen* will be subsumed by *norway*, and *bergen* will not be offered anymore in the current context, being removed from all of its facets. Additionally, all resources being tagged *bergen* will be tagged as *norway* as well [MNS12a].

4.3.2. Version 3

In Figure 4.8, the same example presented for version 4 (Figure 4.2) is shown, but in the setting of the version 3 of the interface. In this screenshot, one of the main differences with regard to version 4 can be observed. The faceted navigation options at the left do not show those tags of the facets that are not assigned to any resources of the result set. In the example case, these tags are *kirkenes*, *kristiansand* and *mosjøen* for the first facet, and *april* and *march* for the second facet. As we can see, such tags do belong to the facets as they appear in the faceted suggestions for the resource titled *Sheep*. In this manner, tags that are not attached to any resources and thus, not shown, cannot be removed from the facet.

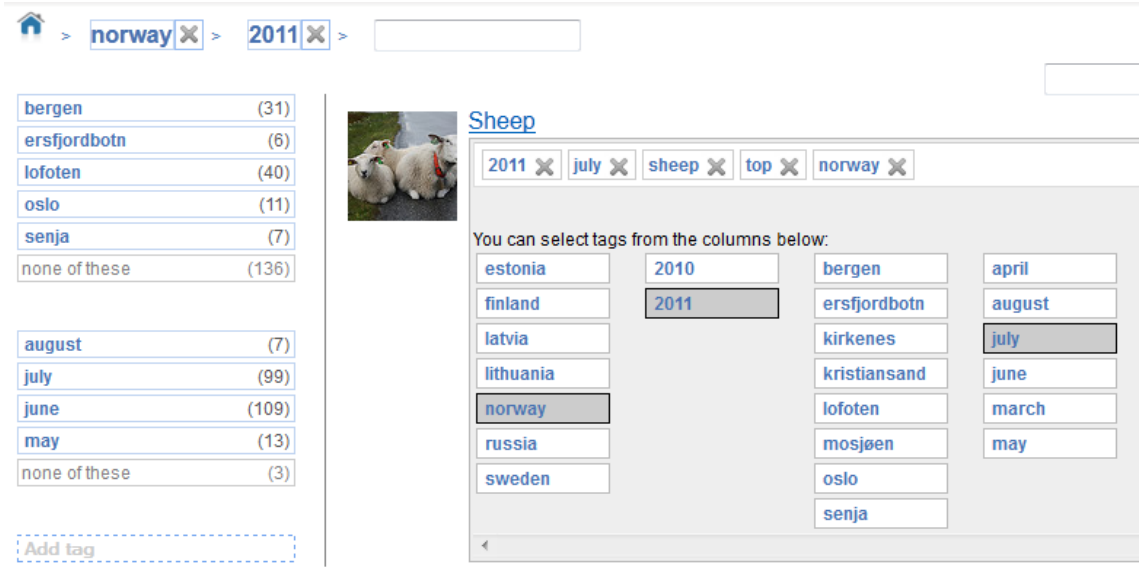


Figure 4.8.: *A screenshot of the same example shown in Figure 4.2 but in the version 3 of the interface. As we can see, tags without matches are not shown in the faceted navigation on the left.*

Probably, the other most relevant difference from version 4 is that this was the last version where in the context of a filter containing negated tags, only the tags from the facet where the label *none of these* had been clicked from were shown. In other words, all the other facets relevant to the same context were hidden.

This version was the first not to introduce previously selected tags on the left side menu, as exemplified in the next section. Also, a set of minor changes were performed. For example, several tooltips were added such as the one for the "add/remove" from all tool. Additionally, the "more suggestions" option was added to support those facets whose potential tags were harder to identify. This way, not only 5 tags where shown, but 20, considerably increasing the probability for the users to find a desired tag. Another minor yet relevant change was the possibility to add several tags to a facet without having to click on "add tag" after adding every tag, so that users could just type the tags, or select them from the autocomplete using the keyboard arrows, and introducing them by pressing enter, one after another, without needing to use the mouse to click on "add tag" every time.

4.3.3. Version 2

Once again, we present the same example in Figure 4.9, but this time in the setting of the version 2 of the interface. It is easy to observe that the facet navigation options of the left menu have slightly varied. Version 2 was the last version to provide the set of previously selected tags on the left menu. This way users could know which tags from facets they had previously chosen. By clicking such tags they deselected them, thus going to the context with the current filter minus the deselected tag. In Figure 4.9 the selected tags are *norway* and *2011*.

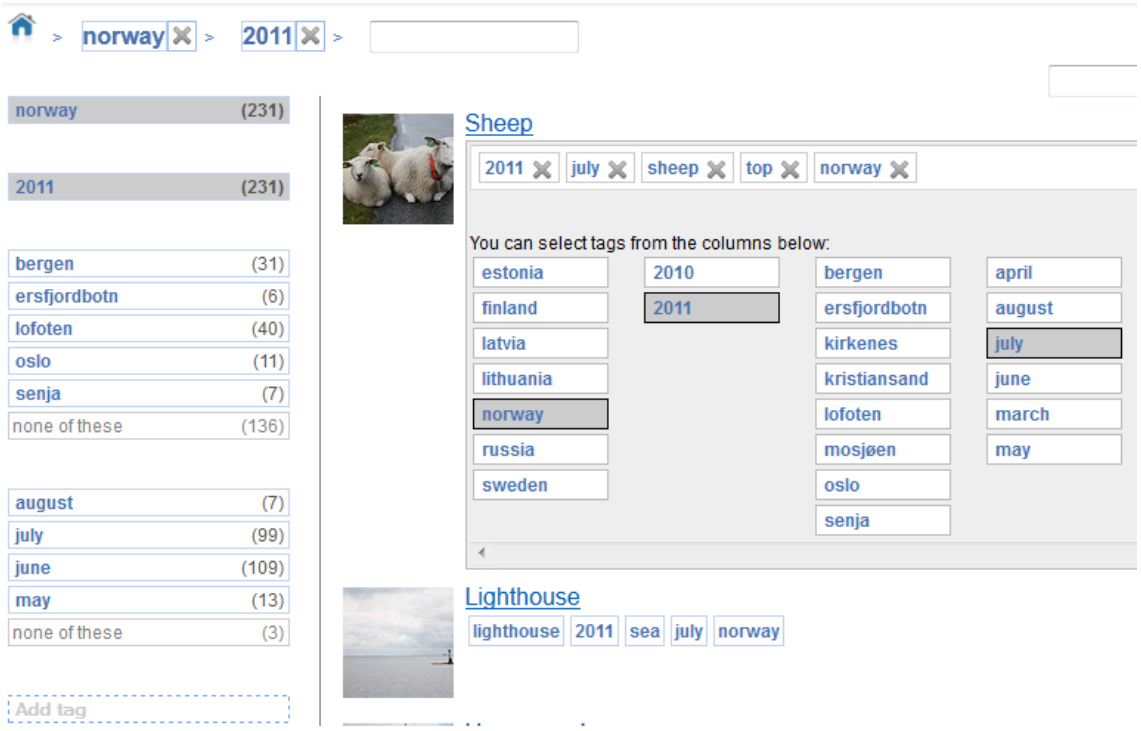


Figure 4.9.: *An screenshot of the same example shown in Figure 4.8 but in the version 2 of the interface. Previously selected tags are shown in the faceted navigation.*

This version supposed quite a major change with regard to previous versions. Firstly, the facet menu on the left side underwent major changes. It was not required anymore to access a facet edit mode in order to manage the facets. This will be explained in more detail in the next section, but it is important to note that this fact supposed a minor drawback, facets could not be directly removed from this version onwards. Also, from this point, facets could be recognized understanding the biggest gaps among tags in the left list as facet separators.

With the implementation of this version, a relevant architectural design change was also introduced. From this point, modifications in a facet, not only affected such facet in more specific contexts of the system, but also in more general contexts containing such facet. In this manner, facets could be managed independently of the context.

Furthermore, the faceted suggestions explained in detail in Section 4.3.1 were added at this point substituting the previous and simpler suggestions.

4.3.4. Version 1

Version 1 is another way of referring to the state of the interface when the first usability interviews were conducted. As we have seen, the interface has undergone several modifications since then.

In version 1, previously selected tags were also shown in the form of a list at the top of the navigation options of the left menu. Deselecting a tag from this list immediately took the user to the new context, result of the current filter minus such tag. This widened the result set.

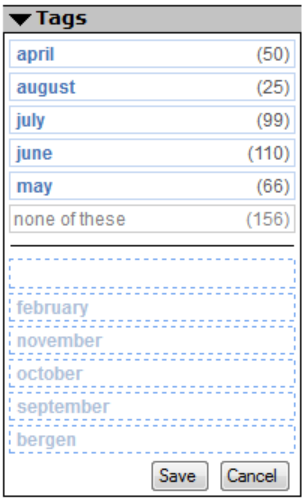


Figure 4.10.: *An example of the edit mode for a facet in version 1.*

Additionally, every individual facet of the left menu offered an edit mode where all the management actions could be performed. To do this, users had to click on a small pencil icon appearing at the top right of the facet when hovered with the mouse. Figure 4.10 is an example of how such edit mode looked. Tags could be individually removed in the same manner as in the other versions, but in addition, the entire facet could be directly removed by a single delete control offered when hovering the top right corner. Such operation required a confirmation. Also, performed changes needed to be saved by pressing the "save" button, or cancelled with the "cancel" button.

As we said earlier, this was the last version where the suggestions offered in the resources edit mode were a simple list of the most frequent tags among other resources sharing tags with the resource being edited, that is tags being frequently used for similar resources.

Finally, it is important to note that at this stage, the design of the system involved that when making changes to a facet in a certain context, only the current and more specific contexts were affected.

Part III.

Tests

5. Plan and Execution of the Tests

After having read about the needed background knowledge on usability evaluation, tagging systems and the concrete evaluated tagging system, we now present all the relevant information concerning the preparation, planning and execution on the tests we conducted for the usability evaluation of TACKO. The study has been performed in both, the context of image search, where the presence of tags is known to be very valuable, and in the context of bookmarks. The resources for such purpose have been retrieved from Flickr.com[1] and Delicious.com[2] respectively, two of the main representatives of tags on the web. First, in *Preparatory Work* we briefly present the development of the interface presented in Section 4.3. In *Selection of the Usability Evaluation Method* we provide a discussion on the chosen methods for this work, providing also the main reasons for such choice. Finally, in *Tests* we present the used dataset, the participants of the test and how they have been grouped for the development of certain tasks.

5.1. Preparatory Work

For the execution of our usability evaluation, we first needed to develop a simple application that incorporated TACKO. In this manner, we created a plugin for the commercial web-based enterprise collaboration platform called Tricia, that allowed to define the concepts of facets on tags as well as their subsumption relations. This plugin is basically a simplification of the TACKO prototype for a better evaluation of its usability, ignoring some of the features the original prototype incorporates, such as the search bar, the diverse view options on the result set, etcetera.

We first created the importer of resources of Delicious.com so that we could work with their bookmarks. Concretely, Delicious.com enabled us to create an importer for the resources and tags of an specific user provided his credentials. For such purpose we used the API from Delicious[3] together with a third-party Java library for Delicious called DeliciousJ[4]. The selection of an already implemented Java library for Delicious.com was not easy because all of the most popular ones were already obsolete due to the changes such website has undergone in the last year. Also, during the timeframe of this work some changes were made that needed to adapt the code to the latest version. As they say in their API page, the API can be subject to changes that may deprecate it agressively, and this is exactly what had happened to the DeliciousJ.

[1] `http://www.flickr.com` (Last accessed on 18/08/2012)
[2] `http://www.delicious.com` (Last accessed on 18/08/2012)
[3] `http://delicious.com/developers#title1` (Last accessed on 21/08/2012)
[4] `http://deliciousj.sourceforge.net/` (Last accessed on 21/08/2012)

Nevertheless, it was still considered better to adapt such library instead of implementing it from scratch. Part of this decision was made because the DeliciousJ library provided some mechanisms to deal with the Delicious API limitations such as waiting at least one second between queries, to avoid getting automatically throttled, or setting the User-Agent to something identifiable, avoiding the default "Java/version-name". For example, not doing the latter at the beginning got us banned for 24 hours a couple of times.

The most used call was *https://api.del.icio.us/v1/posts/all?*, which fetches all the bookmarks by date or index range. Such call provides the possibility to use a set of arguments such as which tag to filter by, the definition of the amount of desired results, a timeframe in between which the resources were added or also a meta field that indicates changes in the resources fields. An example of an HTTP Request looks like this:

$ curl https://mezod:passwd@api.del.icio.us/v1/posts/all

And an extract of the returned data would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<posts tag="" total="268" user="mezod">
<post description="Viquipèdia" extended=""
      hash="705f598341684f756269deae60916433"
      href="http://ca.wikipedia.org/wiki/Portada" private="no" shared="yes"
  tag="reference catalan wiki wikipedia enciclopedia català"
  time="2012-04-08T18:59:11Z"/>
...
<post description="The Structure of Collaborative Tagging Systems" extended=""
      hash="debc8aaeda8cc545af8acd46668491ff"
  href="http://www.hpl.hp.com/research/idl/papers/tags/tags.pdf" private="no"
  shared="yes"
  tag="PFC munich social bookmarking taxonomy collaboration research tagging"
  time="2012-04-08T11:34:51Z"/>
</posts>
```

This data is then processed and its fields can be separately obtainted by the use of several methods provided by the library.

As the process of creating resources inside Tricia (explained below) out of the XML is considerably slow —it may take around 10 seconds per every 100 resources— it is important to note that the system keeps the date of when a user last logged in, so that the next time he logs in, we only need to update his collection. For the update of this collection it is enough to provide the date from which you want to retrieve new resources from Delicious using the same call as before. The forementioned detail is quite relevant if we consider that the importat of resources is automatically executed when a user logs into the system.

Below, we will present the main structure of the import handlers without entering into too much detail but commenting some of the necessary decisions taken due to the architectural design of Tricia. It is important to note that this structure is very similar for the three importers implemented for this prototype: the one for Delicious bookmarks, Flickr photos and Flickr group photos.

1. First, the interface provides the users with a login form where they are asked to introduce their Delicious or Flickr credentials. This way, we do not require the users to previously register to Tricia in order to facilitate the tests. However, Tricia requires of a user (Person) to be created in order to be able to attach the imported resources to someone and also, to define the proper access rights. Therefore, if it is the first time a user logs in, we create a Tricia user account transparently to the user but in relation to his credentials, so that the system will identify the user each time.
2. Secondly, as Tricia works with resources and tags in the context of a wiki, we also need to create a wiki transparently to the user and to which we will assign the user as owner, and its resources.
3. Additionally, we define two local session parameters to be able to identify the user from other handlers as well as to know which of the three importers we are using, a distinction we will need to do at the time to create the path in the future.
4. Once we have the user and the wiki we are going to work with, and after saving the date as "last login" for further updates, we proceed to do the import/update.
5. The import basically consists of a loop that analyzes the returned data of the call to the different APIs (as shown in the example above), which stores each separate resource to the system with their respective fields. For this purpose we create an object, *Assets* in Tricia, for each resource type. Generally, the libraries provide methods to retrieve the different fields separately, so this step basically consists in calling these different methods for every newly created asset within Tricia, that later needs to be persisted. It is important to note that the several tags of the resource need to be treated in a special way, defining them as *Taggable* within the system. The processing of such collections of resources is generally slow. To get a better idea of this it is enough to know that for a collection of almost 300 Delicious bookmarks, approximately half a minute is needed. Similarly, for the import of 50000 Flickr photos, between 30 and 45 minutes are required depending on the machine.
6. Finally, we just need to create a *Forwarder* to the Handler that will actually show the proposed TACKO interface with the already imported resources.

The implementation of the interface basically consists of an adaptation of the original prototype and therefore most of its features are reused and adapted. Similarly happens with the different javascripts and stylesheets as we need to keep them as similar as possible to the original version for a more usefulness of our usability evaluation.

However, it is important to talk about the construction of the path as it is also a relevant source of concerns for the end users. The paths in Tricia require to specify an SpaceFilter, in this case our newly created wiki, a ContentTypeFilter, to distinguish for example Flickr photos from Delicious bookmarks, together with the set of tags users may have chosen to filter the context with, and in case of using pagination, also the page in which the user is at every moment. All of these, encoded with several especial characters that normally result in paths looking like the example below:

```
.../showTags?path={%22all%22%3A[{%22contentType%22%3A%22deliciousBookmark%22}%2C{%22space%22%3A%22%2Fwikis%2Fmezod%22}%2C{%22tag%
22%3A%22webdev%22}%2C{%22tag%22%3A%22programming%22}%2C{%22tag%22%3A%22framework%22}%2C{%22tag%22%3A%22opensource%22}]}
```

5.2. Selection of the Usability Evaluation Method

In Chapter 3, we provided a quite wide overview of the main usability evaluation methods. It is apparent from the list that methods are intended to supplement each other, since they address different parts and tackle usability evaluations from different perspectives and also since their advantages and disadvantages can partly make up for each other. Therefore, it is normally recommended to rely on multiple complementary methods.

At the time to choose a method, it is also important to consider the number of users that are available. For example, for reduced numbers it might make sense to apply heuristic evaluation, thinking aloud and observation while for bigger groups of users, logs analysis, focus groups or questionnaires might make more sense. As it was decided that the interface for TACKO was not mature enough, it made no sense to test it in a large scale as most of the main issues with the interface could already be identified by a smaller number of users. Therefore, methods such as questionnaires or logging analysis, which provide quantitative data were discarded. We initially counted on a group of 5 persons, but due to the constant improvement undergone by the interface, thus involving relevant changes to the interface, new unspoilt users were required to try every new version.

Considering that the prototype for TACKO was already in a quite advanced stage of its development, it was also important to use methods that were especially applicable for advanced or final versions. Also, the expertise of the evaluator had to be taken into consideration, and despite the expertise on how to use the system could be considered medium-high, the expertise on how to apply such usability evaluation methods was considerably low. Therefore, it was also important that the chosen methods were easy to learn, easy to apply, and that the expertise of the evaluator did not have a great impact on the outcome of the observations. Such consideration was important to agree that both, Thinking Aloud and Field Observation matched our needs considerably, as they both are probably the simplest methods since they leave most of the work to the users, while leaving the evaluator silent and observant. Also, by participating in a number of observations and thinking aloud evaluations, a usability evaluator can build up an understanding of usability principles that will significantly improve that person's performance at the time to apply heuristic evaluations. Such understanding also helps the evaluator to become a better interviewer and to design interviews that probe important usability issues.

Therefore, it was found that the application of Interviews, Thinking Aloud and Field Observation combined as in the more general idea of Usability Testing with users, was applicable and suitable for this work. It need not be said that all these test methods can be applied by a single evaluator.

Also, with the knowledge accumulated during such interviews and with the support of several papers related to usability applied in tag presentation techniques and faceted organizations a basis for an informal heuristic evaluation was founded. This way, the evaluation was not left alone with the application of multiple usability test methods, but also included the application of a usability inspection method. Some of the papers that can be considered as heuristics for the matter of tag-based, faceted or hierarchic knowledge organization systems are: [Hea08], [HK07], [YSLH03], [LZT09], [EHS+01], [Hea06b], [Fag10],

[TLP+12], [SvZ10]. Using the conclusions of such papers as heuristic principles, we could identify some of the issues of the interface that would not have been found only by the application of test methods. Also the application of this method was possible because it is very intuitive and it does not require any special preparation to apply it. It is more about *common sense*.

There are two major reasons for the usefulness of combining heuristic evaluation and any of the test methods. First, an heuristic evaluation can eliminate a number of usability problems without the need to *waste* users, who are generally difficult to find and schedule. Second, as explained in Chapter 3, these two categories of usability assessment methods have shown to find farily distinct sets of usability problems, in the manner that they supplement each other rather than leading to the same findings. In other words, the findings of one method are not necessarily ratified by the application of a method of a distinct type.

The application of some aspects from Field Observations was made possible with the help of new technologies that offer the possibility to share desktops among computers so that users' movements around the system can be observed and even recorded without observers breathing down their necks.. The application of such method was helpful to observe new applications of the features, or how users approach a solution in a manner different from which the designer would have expected it. Also, the possibility to observe it in real time gave the evaluator the possibility to ask for ununderstood movements made by the users before they could forget about what they had done and why they had done it. Of course, we are not talking here of the strict application of the Field Observation as communication between tester and evaluator was constant.

A smooth version of Thinking Aloud was also applied. We say smooth because Thinking Aloud can be especially stressing for users who are not used to participate in usability evaluations, and this has been proven to be an important reason to undermine the interest of the participants. Therefore, it was used during the application of brief tasks or in alternative ways where the evaluator also participated trying to be as little as intrusive possible. However, the application of alternative methods based on Thinking Aloud, the also presented in Chapter 3 Constructive Interaction was found way more valuable. Participants working collaboratively to try to understand the system were way more efficient for both, the amount of provided data for the evaluation in the form of verbalized thoughts and also for the participants to understand what TACKO was actually about. Normally, a disadvantage of Constructive Interaction is the difficulty to train pairs of testers to work together, but considering that a big majority of the testers used during our evaluation, who have studied or are in the latter stages of studying informatics, a field where teamwork is greatly enhanced, it proved not to be a problem, but an advantage. Also, with the use of this method an important observation was made clear, the success of TACKO strongly depends on the ability users have to interpret organization structures made by others and to learn from more advanced users in order to benefit from the system themselves. Of course, users should always try to adapt the system to their needs, but it is by seeing how other more advanced users use the system that they will more easily grasp the ideas behind TACKO.

Finally, Interviews were used as the connecting method for all the previously mentioned methods. The interviews could also be done online in parallel to observation with the

help of online voice services and proved to be very useful at the time to focus on specific areas of interest of the users, enabling to improvise new questions disregarding others previously prepared. The problem with interviews is that if not thoroughly controlled, it is very easy to beat about the bush and getting into so much detail in certain aspects of the interface while ignoring or passing by many others that the resultant interviews do not look alike at all, making the inference of conclusions quite difficult if not impossible. Also, the recording of such conversations was possible providing the possibility to go through them again in the future. Again, it need to be said that interviews and especially revising recordings are very time consuming practices. Additionally, the difficulty to arrange suitable times with the users for the development of the evaluations was notorious, especially in the case of Constructive Interaction, where different users had to agree on a date. Thankfully, online tools like Doodle.com, made the scheduling with several testers easier.

5.3. Tests

For the development of the tests two different servers were set up with the latest version of the prototype. These servers were mostly used by the testers to try a little bit the system before actually having to take part in the interviews so that they had enough time to learn the basics by themselves without any external pressures. This was not done like this from the beginning, but after few tests it was found necessary because starting the interviews without users having previously seen the interface could lead to misunderstandings or misinterpretations on the interface. For example, if a user was asked to add a tag to a resource he would try to deliver a solution as soon as possible, and thus, without much thought after it, i.e some users would try to filter the result set to the specific resource in order to use the "add all" feature, instead of previously hovering on the resource to search for an edit mode, only because this did not appear in the interface at the time the request was done, and they had not had any previous experience with the interface. Of course, users would immediately state "this was not obvious" or "this was quite hidden" which, when backed by several users, would lead to the mistake of taking such observation as true. On the other hand, when users had previously had some time to play with the interface, even if only to get a basic grasp of it, options such the edit mode for resources when hovering over them were found obvious and perfectly operational. Nonetheless, about half of the participants never used this servers because of these being offline or simply because they did not find the time to.

For the interviews and application of other methods, simply referred to as interviews or sessions from now on, the following environment was set up: we ran a test instance of the prototype on our local machine, so that it would run faster and the users could be ensured that no matter what they touched they would not be making anything wrong. This is normally not a problem for informatics but it is still relevant to tell the testers that they are just playing with a copy that will be deleted after their use anyway. However, this was not always the case and sometimes a staging mode of the prototype was run even if users were not aware, so that their categorizations could be used again. In most of the cases, where users did not want to use their own datasets or they simply did not have one, the import of resources was done in advance, principally for two reasons, the first and obvious, to avoid making the testers wait at the beginning of each session, and the second and not that obvious, to be able to know the credentials of the specifically created account in case we wanted to use the very same account for further tests with other users.

Additionally, all the users were asked to connect to our local machine by means of the remote control desktop sharing software called TeamViewer which enabled testers to see our desktop and enabled us to observe their moves and use of the system, as if we were physically observing them in a terminal. It is important to note that TeamViewer offers a portable version, thus not requiring to be installed, which is always appreciated by the testers. The interviews were conducted online with the use of online voice services such as Skype or GoogleTalk, depending on the users' preference. Additionally, these conversations were recorded with the default recorder from windows 7. Finally, it is relevant to say that Doodle.com served greatly at the time to schedule the meetings, particularly for those sessions in where more than one tester had to take part.

5.3.1. Dataset

The prototype was initially designed so that every tester could try the system for his own set of resources. However, finding users who usually tag in either Flickr or Delicious, or in any other relevant website applying a powerful tagging system, and who would accept to participate in long interview sessions was not an easy task. Also, the pools of tags by most of the users would prove unstructured and would need previous refinement to be able to properly use the strengths of TACKO for navigational purposes and some of its features such as the *subsumption* relations. Furthermore, despite Flickr provided the possibility to retrieve an specific user's set of resources and tags without needing his credentials, the task of refinement was found arduous especially considering the not familiarity of the evaluators with the set of resources. However, it is important to consider that the familiarity of the testers with the set of resources probably has a great impact at the time to understand the main concepts of TACKO. In other words, it is hard for a user to understand the concept a facet represents if the tags are cities from Norway and the users are not familiarized with norwegian geography.

To mitigate such problems, we prepared especially for our study a collection of 506 Flickr photos with approximately 600 unique tags which proved enough to define four levels of specificity and several categorization dimensions. However, it is important to note that prior to this dataset, another of about 300 bookmarks was prepared in Delicious, and some of the first tests were conducted with this dataset. Finally, some of the more interested users were also encouraged to try the system with a collection of about 45000 flickr photos corresponding to a Flickr group concerning Munich. The collection of 506 Flickr photos, which is a collection of photos taken in northern european countries, was tagged thoroughly and refined so that no tags attached to several resources could be considered useless for navigation because they should have been attached to more resources, etcetera. Also, the interface was used in advance to let the system learn some *subsumption* relations that would have normally been learnt by the system with the management of such tags. We would consider such collection as *well-tagged* because all the resources were tagged with at least a location, a month and a year. And later, depending on their nature, more dimensions were assigned, providing a feeling of completeness among the tags. This obviously had a great impact on the tag suggestions and thus made it easier for the users at the time to complete tasks related with organization and categorization of the resources, as they could more easily understand the concepts of facets and easily attach tags defining similar concepts to them. This proved the importance of suggestions.

Finally, for those tasks related to search or discovery, a complex enough for beginners lattice was built. It basically consisted of the definition of facets representing concepts such as *country*, *year*, *month* or *city*, and with different levels of categorization like the ones shown in the examples of the interface in Section 4.3, where a hierarchy for facets defining the following concepts could be defined (facetConcept:tag): *countries:norway* $\rightarrow$ *places:Lofoten* $\rightarrow$ *islands:vesterålen* $\rightarrow$ *towns:Nyksund*. Obviously, several different additional facets, applying to different contexts could be created such as the ones representing the concepts of meteorological conditions, transport types, places of interest, photo styles, fjords and other geographical features, building types, tags representing opinions, and so forth.

5.3.2. Participants

Our study involved 16 participants, of which only one was a female, who were recruited by personal petition, which means that they are all acquaintances of mine. Sixteen participants was found quite enough even though Nielsen suggests that 20 users is sufficient [Nie06]. All the participants are in one way or another related to the discipline of computer science. Four already hold a master's degree while other 8 are soon to hold it. One of them has earned a bachelor's degree in informatics and another one in business administration though he works as a web applications developer. The last two also work in the field of application development, one in the scope of android and the other one being an specialist in javascript. Their ages ranged between 24 and 30 years old. It need not be said that all the users work with computers and that they use it several hours a day, being all regular users of the internet and searching information (both links and images) every day. Although all of them reported knowing the concept of tags and other related concepts such as tagclouds, less than half of them (7) reported tagging habitually —weekly— and only 4 of them reported having an account in either Flickr or Delicious.

Working with participants who are interested in the system as well as in the collection in question is known to be especially important in usability studies but not so easy to accomplish. The level of commitment ranged notoriously among users, but it has to be noted that in general they were willing to help and provide useful feedback.

As expressed before, all of the testers were acquaintainces of the evaluator and thus, the tests could be conducted more informally, so that users did not feel tense with the situation and helped them to feel more comfortable. Such fact is considered quite relevant because the outcome of the interviews also strongly depends on the users confidence to express immediate thoughts and other questions or observations that could be considered silly, and that they could choose to keep for themselves otherwise.

It is worth noting that the sample of chosen users does not necessarily represent the immediate final audience of users who will use this system, but they can still be considered a potential final audience. This fact also depends considerably on the future acceptance of the evaluated system and similar faceted categorization approaches.

Finally, for the development of our study, we used the *between-subjects* design, which involves creating groups of participants, each of which tests a different system or interface [Fag10]. We present these groups in more detail in the following section.

5.3.2.1. Groups of participants

As one of the goals of this work was to find out which was the minimum amount of information users need in order to be able to start using the interface appropriately, four different groups were created, whose participants recieved different amounts of starting information and previous indications on the system. The number of participants for each group increased in parallel to the number of new versions that appeared for the interface, with a final number of 4 participants per group. However, this does not strictly mean that each version was tested by at least one member of each group as the versions were released

shortly one after another and the scheduling of the interviews could not always be made in time.

Additionally, each of the four first participants to participate in the interviews, one in each of the groups presented below, did an average of three tests in total, that is, they did not only test the initial version but they also participated in interviews where newer versions were evaluated. It has to be noted that after the first interviews, the groups made no sense anymore with them because they had already experienced the first interface, and thus, they already had an idea of how the interface worked and which was its purpose. However, more than half of the issues found in the interface was with the help of these four participants, not only because they had more interest in the interface and how the system worked but also because with the evaluation of several distinct versions they were able to provide feedback comparing different versions. On the other hand, half of the interviewees provided basically quantitative data rather than qualitative, those were useful to confirm some hypotheses but would not considerably help to the discovery of new issues.

The four groups defined with four participants each are:

- **Group 1**: The participants of this group knew nothing more than that they were trying a new approach of tagging system. That is, they were initially not taught with the concept of facets nor explained what could the system be used for. Also, no initial categorization was presented to them. They were presented the system as it is after the importat or resources, with a refinement of the dataset but without any defined facets. The idea for this group was to realize what can users understand without any background knowledge on the system, and what is totally unachievable for them. The members of this group were gradually helped, generally by the response to concrete questions and sometimes with brief explanations on how to use a certain feature.

- **Group 2**: The participants of the second group also knew nothing, but in this case, they were presented with an already defined categorization, a set of well defined facets so that they could easily grasp the idea of what was going on with those groups of tags. The main idea for this group was the observation of how helpful were already defined concepts at the time they first played with the interface. In a way this was to replicate the situation users would find when joining a site with TACKO as a collaborative tagging system.

- **Group 3**: The third group of participants was previously briefly presented the idea of TACKO and other information found relevant, such as the concept of facets and the purpose of faceted classification. Also during the interview and the execution of tasks, necessary information would be provided less reticently. However, this group was provided with a raw dataset, which besides the necessary refinement for TACKO, did not count with previously defined facets and hierarchies.

- **Group 4**: The fourth group was the conjunction of groups 2 and 3. Testers from this group would not only be presented an already defined lattice of facets but would also be explained what were they thought for and how could one benefit from them.

Finally, it is worth noting that all the information previously given to the users of groups

3 and 4 was uniquely regarding to the concepts involving TACKO and not about how the interface was supposed to be used, i.e if tags could be dragged, etcetera.

5.3.3. Tasks

For the development of the interviews, some tasks targetting the three different main purposes of TACKO that we wanted to evaluate, search, organize and discovery, were designed. To consider that the performance of a user was or was not successful is entirely subjective, though based on what was observed. Normally, we would categorize a task as successfully realized when the user found minor or no problems at all. If the way to realize a task was considered inefficient, or the user took too much time, it was then considered achievable but with problems. Finally, if the users wrongly used the system or they simply had no clue on how to proceed, the task was considered unsuccessful.

5.3.3.1. Search Tasks

Search is probably the most relevant and common type of action users perform in tagging systems. Therefore, it is important to evaluate the complexity of searching in our system. For such purpose, we developed a task that had three levels of difficulty. It basically consisted of presenting a photography of the dataset to the users and ask them to find it in the system. After some initial tests, we learnt that we had to forbid users the use of the breadcrumb input, as they tended to use it as traditional tag filters, even if they did not really know which tags to retrieve the image with. Later, in some cases, it was needed to explicitly ask users to use the navigational options on the left.

Figure 5.1.: *An example image of the third level for image lookup search tasks.*

The three levels of difficulty were represented by three different images: the first level was an image of St. Basil's Cathedral in Moscow, easily identifiable so that users generally knew where to find for it. The second level was an image of the norwegian Geirangerfjord. Users, of course, generally did not know its name but the image showed one of the most famous scenes of the norwegian landscape and everyone had one way or another seen it before. Finally, the third level is defined by an image of a nature location unknown to the user. The image, shown in Figure 5.1, provides enough elements to tag it properly, without users being able to use tags that greatly narrowed down the result set.

Another similar task that was harder to apply was to ask users to find pictures complying with a set of restrictions. The way these restrictions were asked to the user was complex, because we had to try not to name any of the tags that would easily lead them to a potential answer. Some basic examples are "find a picture where two animals are looking at you" or "find a very modern religious building". These alternative task would generally require users to navigate through facets at another level, requiring to recover from false steps, etcetera.

5.3.3.2. Organization Tasks

For the organization task one task existed that could be taken to different levels depending on the available time. It would basically consist of asking users to define a concrete facet, even if the concept of the new facet was not proposed. This task helped us to get a better idea of how did users decide when and where to define a new facet. Different strategies applied here.

When possible, i.e for members of the first and third groups (they had no previous facets), a simple organization of the full dataset of 506 pictures was requested. This approach enabled us to understand better how do users understand the system and which relations do they define between tags. Also very curious behaviours were discovered here.

In some cases, other users would perform search tasks in organizations prepared during this task. This was concretely done four times, as a preliminary task for a Constructive Interaction session.

Alternatively, some minor organizational tasks were developed in order to use the "none of these" labels. This tasks were developed generally by users from groups 2 and 4, where we had previously had the chance to prepare the collection. We basically untagged some resources —that were similar to other resources in the system, so that missing tags could be inferred— and asked users to arrange certain facets so that there were no unrelated resources for the concepts of that facet. In other words, we would delete the tags representing countries from approximately 10 images, and ask users to fix it. In this manner, users would need to use the none of these tool as well as search for similar resources to find the appropriate countries.

5.3.3.3. Discovery Tasks

Discovery tasks, or maybe more commonly known as exploratory search tasks, were only applied with testers who would dedicate more time than average. Users were encouraged to explore the resources of the interface and choose a favourite picture. This task showed the behaviour of users when they are not concretely looking for something. For example, the use of the breadcrumb input makes no sense here.

5.3.4. The Process

The undertaken process would vary considerably depending on the available time and commitment of the testers. However, a general structure can be summarized as follows:

1. The users were scheduled for a certain date and provided with an online version of the prototype in case they wanted to get familiar with it in advance.
2. We welcomed each participant with a brief introduction on the purpose of the test, the definition of the several participant groups and to which group they belonged.
3. When it applied (for groups 2 and 4), a very basic initial idea of TACKO was provided and the concepts behind it. Questions were replied but we provided no information on how to perform specific tasks nor justify the behaviour of the features.
4. Then we left them approximately 10 minutes to play with the interface if they had not done it before, in order to avoid misinterpretations such as the ones previously mentioned in Section 5.3.
5. Depending on the group a different set of questions would be asked as for preliminary impression on the system.
6. A quick walkthrough, possibly with examples, around the interface would be performed, especially to guide the uninformed groups, while asking for general questions concerning the interface design.
7. Then the users would be asked to perform some of the tasks presented above in the following manner:
    - Every user from groups 2 and 4 would perform at least one Search Task.
    - Depending on their success they would reach up to the third level of the Search Task.
    - Every user from groups 1 and 3 would perform at least the simplest of the Organization Tasks.
    - Every user from groups 2 and 4 would perform a task involving the use of the ”none of these” label.
    - Especially commited users would perform advanced lookup searches, organization from zero of the 506 images dataset, and a discovery task.

8. During the execution of some of these tasks, users would be constantly asked for the reasons of their choices or asked to verbalize what they were doing and why.
9. Depending on the users, a discussion of certain aspects of the interface would take place, especially when those were asked for found issues, disliked features, or simply "What would you change?".
10. Depending on the users, advanced examples would be shown that might involve the use of the bigger dataset.
11. Finally, it was time to go through the recordings and notes.

Questions were initially thought to systematically go through all the features of the interface, but it was found later, that normally users would have the initiative to comment on those that concerned them or proved to be a difficulty for them, while having to take for granted that some other features had been understood or were of no interest for them.

Alternatively, in two occasions, a Constructive Interaction process was developed for users of group 4 as follows:

1. The users were scheduled with one hour of difference for a certain date and provided with an online version of the prototype in case they wanted to get familiar with it in advance.
2. The first user, who had previously had experienced the interface, was asked to do a categorization of the dataset.
3. One hour later, a new user was requested to do the same with the other user as observer.
4. When he was finished, the first user was asked to perform a couple of tasks on the organization of the second user, who now observed.
5. Then, the second user is asked to do the same with the organization previously done by the first user.
6. Finally, a new organization proposed by the evaluator wass presented to both users who were encouraged to perform more advanced tasks collaboratively.
7. A final discussion took place to comment the different approaches the users might have applied and also on how the behaviour of one user can affect the other one.

Testing the system with multiple users at the same time proved very enriching and also more relaxed for the users who tend to feel they are on the spot, and that they are the ones being evaluated rather than the interface.

Such processes were executed a total of 23 times, 4 for the first version, 6 for the second version, 6 for the third version and 7 for the fourth version. This represents a total of 23 different interviews with 16 different users. The duration of the interviews varied notoriously between 45 minutes and 4 hours, with an average of about 90 minutes. This means approximately 35 hours of recordings.

6. Analysis

In this chapter, we present the analysis and the results of our study together with a discussion of some relevant considerations. To do so, in *Features* we go through the interface feature by feature, for each of which we present the found issues and its causes together with possible improvements or observations that should be kept in mind. In *Observations derived from the tasks* we present some other observations that we could not fit in before, to show how some features are understood by the users or how they have used these features alternatively from our original expectations. A list of the found usability problems for the different participants is summarized in *Summary of the usability problems*. Furthermore, the main found issues are listed in decreasing order of importance in *List of main issues*. Finally, some possible improvements are listed in *List of possible improvements*.

6.1. Features

Before we start analyzing the different features of the interface, some general observations can be introduced. First, it is important to note that all the inputs our prototypal interface incorporates are expecting tags to be introduced. The absence of a search bar or the inability to add a description to the resources, among others, means that the interface is uniquely dealing with tags and this is something users realize quite early. This fact can inadvertently affect the output of this analysis in some aspects. For example, even though users are not immediately aware of this fact, in the future they take for granted that all the inputs are for tags, making it hard to evaluate as to which extent a tooltip is necessary.

Strongly related to the previously mentioned observation, some features are not understood at first sight but learnt after a first use, and even though we will report them as usability issues, they can be considered as appropriate solutions because they do not really involve an important drawback and they never affect the proper understanding of the underlying system.

Obviously, the system has to be consistent and coherent not only when features that affect the current state are applied but also how the same state is shown when accessed in different times. In other words, changing the filter has to immediately affect the list of shown resources but also this list of resources has to be shown in the same order when the same filter is applied in different occasions. Some relevant issues have been found regarding this aspect.

Finally, the diverse autocomplete options offered in the interface might need to offer different pools of tags depending on where they are.

6.1.1. Breadcrumb

To begin the analysis of this feature with a positive observation, we must say that the location of the breadcrumb using all the width of the page is very appropriate for two reasons. The first and obvious is that the set of tags represented in the breadcrumb can be bigger without requiring an extra line. This fact is important for the simplicity of the breadcrumb as a navigation tool, even though it is unavoidable to get a second line for filters including several tags (normally between 8 and 12 tags depending on their lengths). The second and most important reason is that positioning the breadcrumb over both, the facet menu and the result list, is a way to inform the user that any changes in the context represented by the breadcrumb will affect both of the features below. Normally, users understand this by inverse reasoning, once they see a left menu they immediately expect this to be invariable and therefore, they expect only the resource list to change. Some have even commented "the breadcrumb is in the wrong place" before realizing that the breadcrumb also affects the behaviour of the facet menu. This is an important aspect for users to understand that the navigation options are dynamic.

Probably, the first possible issue users meet in the interface is that they interpret the breadcrumb's input as a search input. Normally, users will introduce a single word, which in case such word exists as a tag in the system will be successfully added to the filter. This behaviour is not what users would expect but they can deal with it, because their input is now transformed into some representation they immediately associate with the concept of tag. In the rare case users introduce more than one term, the system reacts the same way, encapsulating the terms into a single tag[1], and thus, users understand the input should be a tag. This behaviour was enhanced in our tests because of the fact that Flickr does not allow tags with whitespaces, and consequently, no tags were tagged with whitespaces. In conclusion, users learnt the input should be a tag after their first try. However, it is recommended to add a default set value such as "add tag" or "filter by tag" that indicates this fact. Introducing concepts like "filter" into the interface might be interesting for users to have a better idea of how the system works.

The representation of the set of tags as a breadcrumb is immediately understood by the users and consequently they know in advance how some actions will affect it. Providing a navigation tool they are familiar with is much appreciated. Concretely, they know that clicking on an specific tag will remove all the tags appearing after it to go back to a more general context. Similarly, they know that clicking on the home icon will remove all the tags in the breadcrumb to take them to the context where they started from. Conclusively, no tag highlighting to show which tags will be removed is required.

An observation that led to the removal of selected tags from the facet menu in version 2 was the fact that users generally use the breadcrumb to deselect tags. The always shown "x" icon next to the tags, which by the way is not being highlighted in the current version, probably has a lot to do with this. It can be observed that users generally use the breadcrumb to go back and facets to go further, especially in the context of hierarchies. In cases where the tags for further refinement are already known, the users will prefer to type

[1] TACKO enables tags with whitespaces.

them even if they are already contained in one of the facets. In other words, if they use the tags in the facets it means that they did not have a clear idea of where did they want to go in advance. From this observations we can also conclude that it is important to offer the possibility to quickly add several tags to the filter without needing to use the mouse to focus the new inputs after every tag addition for example with the use of JavaScript's autofocus. Another interesting observation would be that it might be interesting to prioritize tags contained in any of the facets of the current context in the autocomplete. If users already know where they want to go, is very probable that the tag will have a relevant role in such context and therefore, it is probably defined in any of the facets of such context. This way, manually adding tags would be also enhanced. Nonetheless, other tags must still be offered in the autocomplete because users might want to filter by any other. In any of the cases, the autocomplete should only show tags having matching results in the current result set. In other words, the autocomplete should not offer tags that will lead the user to a cul-de-sac.

The most significant issue found with the breadcrumb is that it involves some differences from the traditional designs. Breadcrumbs generally represent a hierarchy, and this is not necessarily what happens in the TACKO interface. For example, several tags that define no relations among them can be manually added. This can be a source of confusions and misunderstandings because users, who are unable to mentally represent the hierarchies defined in the system, totally rely in this feature to understand at which level of specificity they are. It is not like they immediately realize the differences, if they navigate through facets, then they see the breadcrumb reacts accordingly as any other would, otherwise, they realize the breadcrumb is not representing a hierarchy anymore but the steps they took to arrive to the current context. The issue here is that, the same way being similar to other breadcrumbs has advantages for this system, it also gives rise to new doubts and misconceptions. However, it is worth noting that users are not generally concerned about this issue, they are okay with not entirely grasping what the breadcrumb represents at each moment while they can achieve their purposes.

For a set of given tags in the form of a breadcrumb, one would expect that navigating through the different facets one could get to the same point without adding any tags manually. As currently not all the tags from the current context result list are associated with a facet of such context, this cannot be done. Even though this does not mean a great efficience issue, users hardly end convinced with that they understood what is going on and it might eventually lead to dissatisfaction with the system.

A reasonable and apparently easy solution would be to mark in a different colour the tags that don't belong to any facet of the context defined by the set of previous tags of the filter. In other words, if at the time to add a new tag this doesn't belong to any of the facets for the current context, it might be interesting to mark it somehow, by the use of a different colour or any other mark, in the same manner negated tags are distinguished from normal tags. This way, users will know that such set of tags defined by the breadcrumb will always take them to the same result set but that it does not necessarily mean such context is reached by the access of several hierarchical relations defined among facets. This solution is related to the possibility of adding an element in the facet menu representing the tags not related to any facet for an specific context that we propose in the facet menu section.

This element should of course be designed similarly to the representation of such tags in the breadcrumb, i.e the same colour.

Nonetheless, the calculation of which tags should be distinguished is not trivial, and also, removing tags that are in between the breadcrumb might alter the colour of the following tags, which adds additional calculations to the system.

Yet another alternative or complementary solution to enhance the awareness of the differences involved in this kind of breadcrumbs would be to add dropdown options to the tags of the breadcrumb that represent other tags from the same facet of the same context. Tags that didn't offer alternatives would represent either facets of one tag or tags not assigned to any facets for that context. This solution would boost the usefulness of the tool but involves the same calculation requirements as the previous proposed enhancement. One thing is clear, both could be implemented at the same time because they require similar operations, making therefore, the breadcrumb to be helpful tool to understand the concept of hierarchical facets. Right now, the breadcrumb does not really support these concepts. A dropdown menu would entail new challenges, for example, for single-valued facets a user would probably want to change the hovered tag by one of the list, while for multi-valued facets, a user could decide whether to exchange the tags, or to add the new tag to the filter.

Basically, the general issue of breadcrumbs is not only that they include unconnected tags but also that they do not offer a way to know from which facet was a tag chosen. Distinguishing the unconnected tags would solve the first issue while the dropdown options would mitigate the second issue. Of course there are alternatives, if facets could be defined with names, as defined in [MNS12a], breadcrumbs could offer such information to the user. However, this approach would involve similar calculations —we still need to calculate from which facet is a tag in a certain context to be able to provide its name—, and would not be as powerful as the dropdown options. Additionally, users have stated that defining names to facets whose concept can be almost immediately understood would add complexity to the system without considerably enhancing the navigation tools. In other words, defining names for the facets would improve the learning curve on a short-term but be detrimental for advanced users in the long run. Also, defining such names involves more categorization work for the users, and unless the users are born categorizers, it would just entail more unnecessary work.

Breadcrumb																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
Uses input as a search bar	-	X	X	-	X	X	-	-	-	X	-	-	-	-	-	-
Need to focus new input when manually adding tags	X	X	-	X	X	X	-	-	X	X	X	X	X	X	X	X
Autocomplete leads users to a cul-de-sac	X	-	-	X	X	X	-	-	X	X	X	-	X	X	X	X

Table 6.1.: Breadcrumb's usability issues

6.1.2. Facet Menu

The faceted navigation options in the left, from now on referred to as facet menu, are the essence of this system. We have divided this section into 6 subsections for a better categorization of the issues: *Interface design and behaviour*, *Structure*, *Suggestions*, *Drag & drop*, *Right button* and *None of these*. However, it is sometimes impossible to assign an issue or observation to an specific category, and therefore interferences among subsections are unavoidable.

6.1.2.1. Interface design and behaviour

Perhaps, one of the main usability issues of the whole system and that can easily go unnoticed appears when selecting any of the tags of the system. As this is one of the most used features of the facet menu we present it here. It basically consists of the interface not reacting when a tag is selected to be added to the filter. This problem is very hard to reproduce voluntarily but almost every single user has suffered it. Users click on a tag and the browser seems to start processing the action but at some point the operation is aborted for unknown reasons and no feedback is given to the user. Normally, one or two seconds pass until users realize it and click again, but it still causes confusion.

Having no initial facets has been observed as a problem for those groups of users for who we hadn't previously created them, even if they know the existence of facets in the system (groups 1 and 3). To begin with, a single "add tag" label on the left really goes unnoticed and doesn't draw people's attention. Generally, while exploring they will try every other feature without investigating what this label is for which is quite disappoiting considering that the main strength of the system lies in this menu. The fact is that no initial facets provide no examples on how to use the menu. For example, users of group 1 would start randomly adding the suggested tags, without creating a useful facet and jumping into some other parts of the interface, giving up in a way. Users from group 3, would follow the same steps, but in the cases where they were patient enough to create a second group they would normally realize that they were defining groups of tags, and that this might be the way to define the facets we had previously told them about. Even though they started mistakenly, they would, if patient enough, find out what the left menu is for. On the other hand, participants from group 2, who had initial facets, had no idea of the concept of *facet* but immediately realized that the left menu was used to define groups of tags representing similar concepts. Also their attention would be drawn from the beginning because at that moment, with an empty breadcrumb, the facet menu would prove to be the most accessible navigation option. The participants of the group 4 behaved similarly with the difference that they didn't even need to do the reasoning. In a way, good initial facets prove to be enough for users to understand what the groups of tags in the left represent, how to use them appropriately is a horse of a different colour.

A simple detail that was found useful was to change the "add tag" text label by "add tag to new facet". It is not only that it made the concept clear but that it also introduced the term *facet* to those users who had never heard it before in this context, basically participants from groups 1 and 2.

Another relevant observation is that concerning the minimum number of tags needed to define a concept. It has been observed that normally between 2 and 3 tags is enough for users to be able to define which concept a facet represents. Therefore, it is very important that the example facets contain at least three tags. Additionally, providing at least 2 facets is crucial for users not only to be able to know that more than one group of tags can be defined on the left menu, but also to see how facets are affected when one of the tags in a facet is selected. In other words, if only one facet is defined, when users select a tag from that facet, the new context doesn't show any facets in the left nor how would they possibly have been affected, and therefore users are in a stalemate position regarding the facet menu. They cannot see the point of facets unless there are two defined groups of tags.

Up to version 3, tags without matches in the result set where not shown in the facet menu. This notoriously simplified the lists of tags shown in the left menu, especially in specific contexts, but also made it way harder to understand the relation the current groups of tags had with the ones seen in the previous context. Facets could change from having several tags to a single one, or even to not appear because they were not represented by any tag in that context, but they were still there and the user had no real means to know it. This behaviour was a stretch for users because made it harder for them to sort out what was going on. Not showing the unrelated tags involved several other problems. When a user tried to add a tag to the facet by typing it, the interface would offer no feedback nor show any changes in comparison to before the addition, which normally meant the user trying to add it again as well as to think this was a system bug. After the second try with similar result, users would conclude with several different ideas together with the feeling of an unachieved work. Very few would infer that the tag is not related to the resources of the current context and noone would realize that the tag was actually added. Additionally, this limitation added up with other problems. For example, up to version 2 it was possible to add the same tag more than once to the same facet. This is a problem by itself, but if the added tags cannot be seen, then the mistake cannot be found nor fixed. And yet another problem, even in the case users knew a tag of this kind is attached to a facet and wanted to remove it, he would have no means to resolve it. This problem was especially relevant in versions 2 and 3, where faceted suggestions for resources where already implemented, and all the tags of a facet where shown. Users could see the tags were there but they could not do anything about it. A couple of them found the alternative of attaching such tags to a resource, so that they became visible, then they would proceed to remove the tag from the facet to finally remove it from the resource. This solved the problem but it is of course nonsense and no user would ever dedicate that much time to solve an issue which is in itself already hard to detect.

All of these reasons, solved with the introduction of unrelated tags from facets to be shown, conclude that this design change was very appropriate. Not only eases the management of facets, it also helps to find out which concept the facet represents (f.e in facets where only one tag has matching results), and maybe, the more valuable enhancement, facets preserve their structure, their dimensions, their appearance and prevalence within the interface, their identity. This fact notoriously improves the user experience with the left menu. Groups of tags do not aggressively change after supposedly related contexts such as those accessed through hierarchies. Also the distinction in colour between tags with matching results and unrelated tags is very obvious and clear for the users.

However, not everything is advantageous. Adding the unrelated tags of facets makes the lists longer, and especially in more specific contexts, where more facets appear and the possibility for tags to be assigned to the resources of the resulting set is lower, the number of unrelated tags can over-populate the left menu. It might be desirable to incorporate the option of collapsible facets, that hide all those unrelated tags of a facet. This option notwithstanding, would only be especially important for very advanced users, who have deeply rooted the concepts of TACKO and who can bear facet menus to be constantly and considerably changing. A couple of users also requested facets to be completely collapsible. The tones of blue chosen for the representation of the tags in facets initially seem very appropriate and visually attractive, users are very happy with it. However, a usability issue appears when hovering over the light blue tags (those that are not assigned to any resources of the current result set). Light blue cannot be read over the highlighting grey. It is very easy to work around this, but it still supposes an issue. It is also true that users generally choose a target tag before they are pointing at it, but a better colour combination could be found.

Everything concerning the distribution and structure of the facets is important. For example, starting a new facet should always behave equally, adding the new facet at the end. This is how the current interface actually behaves but there are exceptions. Sometimes, facets don't get added in the last position but before other facets that might have been defined in more general contexts. This fact, probably a bug, does not always happen and is quite difficult to reproduce. One case scenario where this arbitrarily happens is[2]:

1. Create a facet and assign tags to it in the context defined by the empty filter.
2. Create a second facet similarly.
3. Create a third facet similarly.
4. Access another context by clicking on a tag of the first facet.
5. Access another context by clicking on a tag of the second facet.
6. Within the context of a filter with two tags, add a new facet.
7. The added facet may be placed first, before the inherited facet from the more general context (added in 3rd step).

Changes in behaviour like the one presented above should be totally eradicated. Not only they confuse users and make them question previously learnt features, but also affect the understanding of the facet menu. In other words, if users notice a difference in behaviour, they will try to deduce what can it mean developing hypothesis that lead them nowhere. Concretely, the forementioned issue led users to believe facets created in certain contexts were more relevant, because those were placed first, but they obviously could not sort out when this was the case. Users can be very imaginative, especially if they are familiar with computers and also try to understand why systems behave the way they do.

A very negligible drawback from the disappearance of facet's edit mode in version 1 was that afterwards facets could not be removed at once. From version 2, facets need to be

[2]This sequence tends to reproduce this problem more than half of the times.

removed tag by tag. However, this does not represent a big issue because the act of removing entire facets is very rare. It does not represent a major problem to make some more clicks in very occasional moments. A possible solution to this problem is presented below together with another set of enhancements.

As we have repeatedly said, facets are the soul of TACKO, therefore, it might be interesting to upgrade them if this can suppose a considerable enhancement for the concept they represent. For example, being able to specify a facet name would in some way involve an improvement. However, one of the main strengths of the current version is the simplicity in which facets are represented and we are quite reluctant to change this. Nonetheless, we proceed to present what we could call the *facet header*. This facet header would come to be a div of similar height and width the tag representations have but with an slightly different design as shown in the prototypal design of Figure 6.1.

Figure 6.1.: *An example of the facet header when expanded and when collapsed.*

The facet header would incorporate the following details:

- It would be a platform where the collapse options are displayed. In Figure 6.1, the left representation shows a facet that would not be collapsed while at the right we can see the very same facet collapsed.
- A number at the top left (In Figure 6.1 is a 7) which represents the number of tags assigned to the facet. As we can see this number doesn't change independently of if the facet is collapsed or not. With this feature, users can know if there are unrelated tags even if they are not shown, and by simply expanding the facet, they can know which are those tags. Numbers can be very helpful to help users understand what is going on in the interface, but one should just be careful not abusing of their use.
- In the case it was decided that facets should have a name, this would be a good area where to place it. However, we haven't added it to the example because we don't support such feature.
- A number at the top right (In Figure 6.1 is a 154) which represents the number of matching resources in the current result set. In other words, the number of appearances of the tags of this facet in the resources of the current context. This is not the number of resources of the result list that contain tags from such facet. This number

is the total sum of the numbers next to all the tags of the facet. The idea of this number is not only to provide a better idea of the relevance of the facet in that context, but also it immediately enables us to know if the facet is single-valued or multi-valued. This feature can serve, therefore, to raise users awareness towards the possibility of both types of facets without inconvenient. This is relevant because users seem to have a general tendency to create single-valued facets, i.e without overlapping, probably as a result of familiarity with traditional hierarchies. It is not hard to see that if the total amount of resources in the current context minus the resources represented by the *none of these* differs from this value, there's overlapping and thus, we have a multi-valued facet. If this were implemented in the version 4, where a counter of total resources is included, this would help users to easily realize such fact. Furthermore, these numbers could be coloured differently depending on if they represent one type or another, and a tooltip could guide users to the right conclusion. However, unlike other concepts such as *facet* and *filter*, the concepts of single and multi-valued are not so relevant to the user and might want to be avoided.

- Below, we will discuss about the possibility to reorganize facets within a context. This header could serve as a platform to show the dragging pointer once hovering over it. This would not only show the users the facet is draggable, but also raise awareness in the system for features such as tags being draggable.
- Showing these headers would provide an even clearer and more obvious distinction among facets than the one provided by the current greater separation between facets than between tags.
- A delete feature could be offered similarly to how a single tag can be removed from a facet.
- Last but not least, users would embody the concept of the facet with this header. In other words, they would think that header is the facet. This would for example provide an obvious area on where to drop a dragged tag. Theoretically, indicators would not be even needed because users would already understand they are dropping a tag into somewhere that can contain it.

It not need be said this facet header would solve several problems at once, but it still would be recommended to assess as to which extent this feature would represent an advantage contrasting with the current apparent simplicity of facets.

The possibility to add tags one after another to a facet by simply typing and pressing enter has been found very useful as opposed to version 2, where after every tag it was required to click on add tag with the mouse, notoriously slowing the process. Normally, the use of *accelerators* or hotkeys, such as adding multiple tags separated by enters, is much appreciated.

Moreover, the use of highlighting for tags in both, the facet menu and the breadcrumb, has also proven not to be just decoration, but also a successful way to give feedback to the user of when a tag is ready to be clicked.

A known bug that we still need to list here is the fact that some "add tag" buttons appear in bold blue after having dropped a tag to any of the facets. This occurs occasionally but is

still an issue that should be fixed.

Finally, a notable issue appears when users want to drag tags from the facets to the last resources of a page, or similarly, when a tag of one of the last resources of a page needs to be dragged into one of the top left facets. Facets are statically shown at the left, which once users scroll down might make facets to disappear from the screen or to appear too far up left. Dragging to facets that don't appear at that moment is still possible by dragging and scrolling at the same time, but normally both actions are performed with the index finger, and thus, represents a very unnatural action for the users. The most obvious solution to this problem is to have the facets *follow* the user while scrolling down. Similarly with the breadcrumb, that disappears after scrolling, that could be made located in a fixed div at the top. However, facets following users scroll might present some inconveniences. For example, if the height of the overall set of facets is superior to the height of the browser, then clumsy solutions might be required. It is important to note that the application of dynamic loading of new resources in substitution of pagination will boost this usability issue.

Facet Menu: Interface design and behaviour																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
System aborts tag selection	X	X	X	-	X	-	-	X	X	X	X	X	X	-	X	X
No initial facets	X	X	X	X	X	-	X	X	X	X	X	X	-	X	-	X
Groups of facets are not obvious enough	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Unnoticed add tag to create new facets	-	X	X	X	-	-	-	X	X	-	X	-	-	-	-	-
Adding a tag without matches has no effect (v3)	X	X	X	-	X	X	X	-	X	X	X	-	-	-	-	-
Tags without matches are not shown (v3)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Same tag could be added twice to a facet (v2)	X	X	X	-	-	X	-	-	X	X	X	-	X	X	X	-
Tags without matches could not be removed from facets (v3)	X	X	X	X	X	X	X	X	X	X	X	X	-	X	X	-
Tags without matches cannot be read when hovered over	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Facets getting added in different positions	X	-	-	-	X	-	-	X	X	X	-	-	X	X	-	X
Facets need to be removed tag by tag	X	X	X	X	-	-	-	-	X	X	-	X	-	-	-	X
Tags have to be added one by one (v2)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Dragging tags to last resources of the result set is inconvenient	-	X	-	-	X	-	X	-	-	X	-	X	-	-	X	X

Table 6.2.: Interface design and behaviour usability issues found in the facet menu.

6.1.2.2. Structure

Probably, one of the architectural design decisions that involved more problems at the time to understand the behaviour of the facet menu was the fact that modifying a certain inherited facet only affected the facets of the current and more specific contexts and it cre-

ated a distinction from the facets representing the same concept in more general contexts, which the system interpreted as two different facets. The problem here is that users did not conceive this distinction, or, in other words, users expected facets representing the same concept to always be the same object, and thus, the alteration of any of its fields should be consistent in any other contexts where this facet appeared. A very typical issue was that a user defined a facet, f.e with the tags *russia*, *lithuania* and *tallinn*, then he accessed a more specific context and realized that *tallinn* made no sense in that facet and subsequently fixed it. However, when he navigated back to the more general context, or to any of the other specific contexts of this general one, he found that *tallinn* was still there, meaning that sometimes he had to fix the problem in several facets because he was not aware that fixing it in the facet where he had defined it would arrange it. Also, this was a very normal error for beginners, who still did not really understand the system and had serious problems to know or remember where they had defined a facet. In conclusion, this behaviour did not help users at all, because depending on which context did they go to after editing a facet would appear modified (more specific contexts) or untouched (more general contexts). The modification of this aspect in version 3 involved a great improvement at the time to ease users understanding of facets even when these did not initially behave as they would expect. This design change involved consistency among facets, because now, if a tag disappeared when acessing a more specific context it could uniquely mean that the tag was not attached to any resources of the result set, while before it could have also been that a user had chosen to delete it from the facet. The design in versions previous to the v3 was probably more powerful but an important trade-off between its learnability and the possibility to define differences in facets among contexts existed, and that proved not to be worth. Furthermore, considering that the system is thought for collaborative purposes, it is very important that changes are consistent for facets, independently to which context they are modified from.

It is also important to note that users would tend to desire to delete tags from facets of more specific contexts where these would not have any matching resources left due to the high level of specificity. In other words, they would not mean the concept does not belong to the facet anymore, but that in such context, it is not useful. This problem can be strongly mitigated with the use of collapsible facets as proposed in the previous section.

Maybe, an interesting improvement in this direction would be to somehow distinguish for a certain context, facets that were inherited from more general contexts, where changes would affect in both directions, and facets defined in the current context, where changes would only affect the more specific contexts. This distinction could be made by changing interface design attributes such as colour as well as with the distribution of the different facets within the facet menu. In fact, this would be very easy if the facet menu was strictly structured as proposed below, with a thin line separing facets in these two groups.

Another relevant enhancement, which the system already complies in most of the cases, would be to force for a certain context that facets defined in more general contexts appeared in the first positions and the newly created facets appeared below, so by accessing different levels of specificity such facets always appear in the same position while they still make sense to appear (i.e non of its tags has been selected), so that users do not find a totally new environment after acessing every new context. This fact takes us to reconsider

the possibility to somehow represent the facets with already selected tags similar to the used in version 2. Consistency within the facet menu is important and choosing a tag from a facet makes that facet disappear in the following context. This is generally understood by users, and therefore, it seems a good decision to remove it from the facet menu, especially considering that "selected tags" do not involve any new features, because users generally use the breadcrumb to deselect tags and remove them from the current filter. A better solution is probably to animate the facet menu when accessing a new context. For example, if in context *A* we have the facets *a*, *b* and *c*. And we access a more specific context B by chosing a tag from *b*, *a* should immediately appear in the same place, *c* animatedly move upwards to occupy the room left by *b*, and a possible new facet *d* animatedly pop up. This would provide a lot of useful feedback to users on how the facet menu is built in every context.

All of these structural options aim to minimize the variability of the facet menu, when possible, and together with the now appearing tags without matching results, this could be achieved considerably. Especially, considering that facets defined in the empty filter context will appear independently of the context, unless some of their tags are chosen, but by then, users will have already grasped the basics of the facet menu.

Not showing the selected tags in the left menu has removed innecessary redundancy with the breadcrumb. The breadcrumb proved to be enough to show the users where do they come from, thus minimizing the facet menu. Additionally, in the version 1 the list of selected tags involved minor confusions because it had a similar design to that of the facets placed immediately below.

Another improvement one could consider is the possibility to reorder facets within a context. Nonetheless, concerning what we have previously mentioned about constraining the order and structure in which facets should be shown we discard this option for now. This feature could be interesting for users' personal customization of the interface, but should under any circumstance apply to the general audience of the system, because not so experienced users would find serious difficulties to acquire the concepts and behaviours of the facet menu.

A similar consideration, backed by few user requests, arises at the time to define the order of the tags within a facet. Currently, these tags are shown alphabetically but this can prove to be a very negligible issue for groups of concepts that generally involve an order. A very obvious example are the months of a year. Some users would expect *january* to be the first and *december* to be the last tag shown in such facet, because when users think of *november* they inconsciously search for it at the end of the list, like if it were a calendar. Additionally, not showing such types of concepts in their natural order makes it more difficult to find missing concepts. For example, in our dataset we had no photography taken in *march* and users would see there was a missing month but would need to make an extra cognitive effort to find out which one was missing. However, this does not represent a real problem and users immediately get used to it. Ways such as ordering the facets by number of matching results and similar should not be considered because they would aggressively change among contexts, difficulting the understanding of the whole system.

As we have seen in the previous section, it would be interesting that the facet menu scrolls

down together with the user, so that it is always visible. For this option to be feasible, a restriction appears: The height of the entire facet menu should not be higher than that the screen can show. Now this proves to be a problem because users work from very different terminals and resolutions and it is hard to find a solution for everyone. Features like the possibility of collapsing facets and similar can help. At the moment, the problems arise from numbers starting in 20 tags distributed among 3 facets approximately, generally in 13" screens where the typical resolution is between 1280x800 and 1600x900. In conclusion, the problem of how to control the number of tags per facet and the number of facets per context must be considered.

This problem might disappear with a *good practices manual*. In other words, maybe it is important to study in detail when a facet is useful in a certain context, or when it can be considered to be in the wrong place. For example, facets that represent under a certain % of the result set resources can be found irrelevant in such context. However, this limitation is very tricky, because some facets might contain concepts that quickly narrow down the filter, thus being interesting to incorporate. At any case, what definately comes to light is the fact that users are initially not very sure of where to define certain facets. It is cognitively challenging for them because the concepts defined by facets are normally too general to be easily associated with the collection of resources of the result sets. This involves a great variability in the approaches and strategies used by the different users, which can be a problem when put together into a collaborative tagging system. For example, in early versions where the system was more flexible (and way more chaotic), a quite normal strategy for users would be to define all the facets they could come up with in the empty filter context and delete them in those contexts where they didn't apply or had no matching resources. The latter was generally not being even needed to be deleted because facets without matching tags were not shown, thus supporting the use of this strategy. However, with the architectural design changes involving bottom-up inheritance these practices could not be applied anymore. This still enlightens us with the various categorization styles users might come up with at the time to organize the resources. All of these observations lead to the idea that general common practices should be enhanced in order to heterogenize the diverse strategies the users might adopt at the time to use the system. Limiting its flexibility doesn't necessarily mean its strengths are weakened while it involves a notorious improvement in the learning curve.

A bug detectable in version 3, which did not represent a major usability issue for users because they would not notice it at first, was that if the first tag added to a facet had no matches in the result set, an special facet was created for it, so that the next tag added to the facets was created in a different facet. As unmatched tags were not visible, this was not a problem in the facet menu, but, once the faceted suggestions of the resources' edit mode were consulted, several lone tags appeared that introduced complexity into the feature, in itself already not easy to understand.

Similarly, in v2 a tag could be added several times to the same facet. This was especially troubling when unmatched tags were not visible, because this problem could not be immediately observed in the facet menu, but came to light in the faceted suggestions feature.

TACKO enables to define the same tag in more than one facet for a certain context. Nevertheless, when this tag is removed from one of these facets, the tag also gets deleted from

the other facets. The decision of enabling tags to be attached to more than one facet per context is not very obvious for the users, some have even considered it as a possible bug. A further problem is that, in the case of implementing the dropdown options proposed for the breadcrumb, there would be no obvious way to know which tag of which facet is the one being listed. In other words, it would be harder to know the tags which facet should be listed.

Concerning the autocomplete incorporated in the input field to add new tags to a facet, considerations such as which set of tags should be suggested appear. For example, this autocomplete could avoid suggesting already added tags to the edited facet, or tags that are already in any of the facets for the current context, or it could even limit its suggestions to tags that exist in the current result set. This is under any circumstance considered as a usability issue, and concretely, users agree this is the autocomplete that better serves its purpose in the whole system.

An option that could also be interesting is to add a ”tags in none of these” or ”rest” label at the end of the facet menu that included all the tags with resulting matches in the current context that have not been assigned to any of the facets of the context. This label could include a number indicating how many tags contains. Ideally, all the tags should be assigned to facets in every context but this is of course utopic and not even necessary, because a limited number of facets is enough for categorization. However, adding such a feature would offer the sense of completeness while serving as a representation of all of those tags relevant to the current result set. The new group for unassigned tags, could work similarly to the *more suggestions* option, in the way that when you click it it expands showing the most represented tags (in this case without restrictions), but instead of clicking to add it somewhere, it would let users to drag these tags into other already defined facets or to create new facets. This behaviour would mimic closely the current behaviour for categorization in folders. Nowadays, it is quite common to be inside a folder that has other subfolders defined and a set of resources, and drag the diverse resources into these folders according to certain parameters. This behaviour could be imitated in TACKO at the time to classify tags in facets. Generally, the tags with most occurrences are most likely to be added to a facet, and if this option listed the tags in a drecreasing order of occurrences, showing the number of occurrences next to each tag, it could be very practical to drag those tags to the facets. This idea appears from observing some users who would prefer to first create a ”dummy” facet, full of tags representing several different concepts, to drag them later into wherever they found appropriate. This behaviour was probably also enhanced by the need to click ”more suggestions” after every added tag after the first 5 suggestions. The appearance of this behaviour is quite easy to understand. When adding tags to a facet, there is no way to remove those that you are not interested in, so you have to go through them several times (every time a new tag is added), which considerably slows the process. The fact that the suggestions can change depending on the previously changing tags, obliges users to pay attention every time, and for very concrete tags this can be very tedious. On the other hand, with an expanded list of the most relevant tags, these can already be dragged into their respective facets, thus going through the list only once per context. This problem would not exist in the ideal case of perfect suggestions so that the right tags were always at the top, which is quite unachievable.

Any way, we want to insist in that one of the main strengths of the system is the apparent simplicity of the facet menu, and one should tread carefully at the time to add features to it.

Facet Menu: Structure																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
Changes in facets do not affect more general contexts	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
List of selected is confused with other facets (v1)	-	X	-	X	-	X	-	-	-	-	X	-	X	-	-	-
Tags are ordered alphabetically within facets	-	-	X	X	-	-	-	X	-	-	-	-	-	-	X	-
Creating facets with unmatched tags made invisible one-tag facets (v3)	X	X	X	-	X	X	-	X	-	X	-	X	X	X	X	-
The same tag could be added several times to the same facet	X	X	X	X	-	X	-	X	X	X	-	X	X	X	X	X
When a tag is added to several facets, it cannot be removed uniquely from one	-	X	-	-	-	-	-	-	-	X	-	X	-	X	X	X

Table 6.3.: Structure usability issues found in the facet menu.

6.1.2.3. Suggestions

As we have just seen, the quality of the suggestions has a great impact in the way users utilize the system. At the same time, suggestions are very dependant on the quality of the tags in the system. Therefore, it would be interesting to promote some practices among the users, to improve the general quality of tags, as well as to enable the system to learn the relevant *subsumption* relations. Currently, users do not think of refining the collection of tags because they are not aware of the advantages such refinement could yield to them. This way, the system has no chance to learn from users actions and also provides suggestions limited in quality. Users must be taught in this direction, being presented with how can they benefit from it.

What is obvious is that suggestions when adding tags to a facet help the recall of related concepts, especially when users are unfamiliar with the set of resources, which is normally the case in collaborative tagging systems where everyone can add tags and resources.

At the moment, when a tag for the facet is not found among the first suggested five, the more suggestions has to be clicked to see the next 15 most possible tags. Normally, many desired tags are found among this bigger group. The problem appears when, after choosing one tag, the menu refreshes showing again the first same five tags and the collapsed more suggestions option. As choosing tags below the first five has poor influence on the apparent relevance of the first five, this tags never change independently of how many tags are added, thus providing poor usability. Additionally, after every tag is added, users need to click more suggestions again, which considerably slows down the process. As we have seen, alternatives to avoid this process have appeared. Additionally, there are some cases, especially in big datasets, where 20 suggestions might not be enough, for example

in cases where the most occurrent tags do not belong to any facets and therefore keep appearing as the top suggestions. Furthermore, these tags cannot be discarded in any way from the suggested list, so no alternative is offered to the users.

A possible solution is to offer users the possibility to discard tags from the suggestion list. Instead of a "more suggestions" feature, a list of 5 tags that can be removed from the suggestions list and substituted by the next most possible tag would be quite appropriate. The discarding of tags would also provide useful information to the system for a certain facet, though such information does not necessarily need to be kept. The controls could be similar to the ones for a tag in a facet, clicking on it would be choosing it and clicking on the "x" at the right would be discarding it. This feature would enable users to choose all the tags of a facet going through the list just once, and thus speeding up the process.

Facet Menu: Suggestions																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
Need to go through all the suggested tags every time users are adding a tag	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Need to click more suggestions every time a tag is added	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 6.4.: Usability issues from suggestions found in the facet menu.

6.1.2.4. Drag & drop

One of the most relevant issues of the interface is that users have no way to know about the existence of drag&drop options within the system. Even though this feature is becoming popular in the web, it is still very early for users to expect such features to be implemented and therefore, most of them don't even think of it. What is true is that as soon as users know an element of the system is draggable, they will try to discover new draggable features. Therefore it would be interesting to show that tags are draggable at least in one place of the interface. The problem here arises from the fact that tags can be both selected and clicked, and thus, showing the pointer for dragging options when hovering on a tag could also make the user thing such tag is not clickable. Besides the more than reasonable solution of basically informing the users of what can be done with the use of tooltips, the option of showing the clickable cursor when moving the cursor downwards and the draggable cursor when moving it upwards can be applied. Like for example:

```
$0.addEventListener("mousedown", function () {
  this.style.cursor = "move";
});

$0.addEventListener("mouseup", function () {
  this.style.cursor = "default";
});
```

Additionally, once users already know that tags can be dragged, the indicators of where can tags be dropped are totally overlooked. Basically almost all the tested users didn't

notice such feature, and the ones who did was after having dragged several times, so that the information provided by such indicators was irrelevant to them. This is mostly because at the time users click on a draggable element, they already have an idea of where they are going to drop it and thus, they are only paying attention on dropping it in the area they think will lead them to the correct outcome. Indicators that appear in the short instants while a tag is being dragged have to be way more accentuate and eye-catching if they want to be successful. However, it is worth noting that this does not represent a big issue, because once users know tags are draggable, with a bit of common sense and performing a trial and error process, they will learn it soon.

Another related issue with drag & drop options is that as we have seen in Section 4.3, dragging a tag can mean several different actions. It can mean ”copy into”, ”move” or ”move into”. In the current version, users have no other alternative than to learn it by observing its behaviour, which of course is not obvious. Additionally, undoing some of these actions is not trivial and requires a certain command over the interface, which at early stages is normally not the case. Thus, it is important to offer reliable rollback options.

In the case of ”moving into” (when a tag is dropped over another one in the facet menu), users are totally unaware of what they have done. First, they have created a new facet containing the dragged tag in the more specific context defined by the current context filter plus the tag over where the dragged one was dropped. Few users can get to the conclusion that they have defined a hierarchical relation, especially if they don't know anything from the system from a theoretical point of view (groups 1 and 2). Additionally, such action creates a *subsumption* relation they are totally unaware of. They absolutely have no clue of what they have done, which is a big problem for those tags that were mistakenly dragged into another one, because not only they don't know how to undo *subsumption* relations but also they don't know they have created them.

Facet Menu: Drag & drop																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
No way to know tags are draggable	X	X	X	X	-	X	X	X	-	X	-	X	-	X	-	X
Drag & drop indicators are totally overlooked	X	X	X	X	X	X	X	X	-	X	X	X	X	-	X	X
Drag & drop can mean many different actions	-	-	X	X	X	-	-	-	-	X	X	X	X	X	X	X
Drag & drop a tag into another tag creates a hierarchical relation as well as a *subsumption* relation	X	X	X	X	X	X	X	X	-	-	X	-	-	-	-	-

Table 6.5.: Usability issues from the drag & drop options found in the facet menu.

6.1.2.5. Right button

Similarly to the drag & drop options, users had no real way to know about the existence of this feature, noone found it without being told.

The right button is a feature that was added in version 4 and therefore it was tested by less than half of the testers of this work. However, it was enough to offer relatively concluding

observations regarding the three main actions it enabled to apply:

- **Delete**: Noone used this feature because the other way to delete a tag from a facet, by clicking on the "x" that appears when hovering over a tag, not only was more obvious but it also required less clicks. This option is totally unnecessary.
- **Rename**: This feature was sparingly used but is important to provide it. However, it involves a relevant problem: When the tag name is changed, the page automatically refreshes showing the renamed tags without matching results, another manually refresh is required to see the renamed tag appearing again with the total amount of matching results. Additionally, for tags with a big number of matching results, several seconds might pass before manually refreshing the right amount of tags is shown. It might be interesting to use the counter of the tag prior to rename to immediately show it to the user in the similar way tags from the resources in the result set are exchanged. However, this trick might not be applicable for tags being renamed with the name of another tag in the same facet (like to merge tags).
- **Quit**: Almost noone chose this action. Normally to click out of the menu is preferred and requires less effort from the users.

An important consideration is that some of the users initially expected "rename" to affect only the resources with such tag in the current result set. That rename affects the whole system independently of from which context it is applied, offers consistency and coherency throghout the system, therefore, we consider the current implementation to be good. However, this decision limits the system. One of the greatest challenges users find when testing TACKO is that is initially quite hard for them to understand that from one context they can be modifying many other contexts (like with facets inheritance). Nevertheless, for resources it might make no sense that the system behaves like this, the same way a tag is added to all the resources of the current result set, renaming should affect them. Users want to apply actions to the currently listed resources, and that is why they filtered them.

After observing that 2 out of the 3 options of the right buttons are unnecessary, a redistribution of the actions seems to apply. The left click should of course keep meaning *select* because it is the most relevant and used action. Editing has lately become popular in the web with the use of the double click. For example, doubleclicking on a tag could show it on an input mode, where it can be edited and saved again by pressing enter. Additionally, a certain tag could be negated by the use of the right click. Right clicking on a tag would therefore mark such tag in red and add such restriction to the filter. We present the need of individual negated tags in the next section.

In the case the current rename should stay, it would be interesting to enable keyboard keys to interact with the dialog box, that is, that the "enter" would work to confirm the new name for the tag without needing to use the mouse. Additionally, users would expect to be able to exit the dialog box by clicking anywhere in the screen outside it.

Facet Menu: Right button																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
No way to know these feature is implemented	X	X	X	X	X	X	X	X	-	-	-	X	-	X	-	-
After refreshing a renaming the tag shows as unrelated	-	-	-	X	-	-	X	X	X	-	-	X	-	X	-	X
Rename's dialog box requires to be confirmed with the mouse	-	-	-	X	-	-	X	X	X	-	-	X	-	X	-	X

Table 6.6.: Usability issues from the right button options found in the facet menu.

6.1.2.6. None of these

When in the *context* of a "none of these", removing a tag from the left menu actually affects the facet instead of removing the tag from the restriction. This behaviour is consistent with the rest of the system, but, however, users do not understand the list of tags in red as a facet anymore but as a representation of the restriction. Therefore, they understand that by removing a tag of such list they are simply removing it from the restriction and not from the facet. This leads to several confusions, especially when users do not immediately realize what they have done. This happens because a user might decide to individually "re-accept" several tags for a certain context. In fact, once users realize the power of being able to add restrictions of a negative nature to the filter, one of the most requested features is the possibility to individually add negated tags. In the current system it can be done by defining a one-tag facet and choose its "none of these", but this is just a workaround to a missing feature. If a context filter is defined only by the "normal tags", it should be no problem to enable users to define individual negated tags. The breadcrumb could provide a way to directly add negated tags to the filter. Or maybe to negate them once they have been added with the use of the right button. Additionally, it should be possible to negate tags in the facets. As negating a tag does not change the current context but only the current result set, the facet menu would only vary in the number of matching results of every tag. Moreover, the structure of the facets would not be affected by the negation of a tag, simply that tag would appear in red, in a similar way to current negated tags. This way, different facets could have some tags negated and some others not being relevant to the current filter. Negating a tag from a facet could be done with the right button and to "re-accept" it a further right click would apply. Of course several observations should be taken into consideration. For example, the "none of these" would represent like a hotkey to negate an entire facet. If all the tags of a facet besides one should be negated then "none of these" is clicked and then right click to the specific tag would do the trick. What is obvious is that a way to quickly add individual tags to the filter should be offered together with the possibility to partially negate facets.

Partially negating facets would support the concept of multi-valued tags. In the current version, where an entire facet is negated, it is impossible to assign two tags from the same facet when using this tool, and thus it is enforcing the concept of single-valued facets, which are already preferred by the users at the time to categorize the tags. However, this behaviour is a replica of the use in traditional hierarchical categorization systems, which doesn't necessarily mean it is bad, but with TACKO we want to go one step further. Be-

ing able to individually negate tags would enhance the possibility to define multi-valued facets.

The fact that in the latest versions the possibility to see all the facets of the current context together with the negated facet supports this idea. At the beginning multiple "none of these" could not be chosen but right now there is no limit in the number of negated facets a context can be filtered with. Therefore, why to define one-tag facets to do this when simply negating tags would mean the same result? Users have proven this would not be a problem for them at the time they have applied more complex workarounds to obtain the same results.

At this point, the need to improve the breadcrumb so that it successfully represents the filter appears. It not need be said that adding negated tags to the breadcrumb also goes against its essence, but it doesn't cause any misunderstandings among the users, who find it a good representation to visualize the filter of the current context. Grouping all the negated tags of the filter in the current version is very useful to remove them from the filter in one click. Additionally this can be done in the facet menu by clicking any of the tags of the negated facet. However, it would be very interesting to be able to individually add and remove negated tags from the breadcrumb. Another known issue is that representing the negated tags with the "NEITHER tag1 NOR tag2 NOR tag3 ..." is neither visually nice nor shows all the tags of such filter in the case of long lists. An alternative might be to list the tags similarly to the normal tags but in red. This would enable to easily discard single tags, or to change a "requirement" into a "prohibition" or viceversa. However, for contexts with several negated tags this might require too much room, and the breadcrumbs might be filled with several tags altogether. Another option is to simply add a label indicating negation, probably in red, which when hovering it indicates a list of all the current negated tags in the filter in the form of a drop-down menu. Of course, individual removal options, choosing a tag from the drop-down menu, or global removal options, simply clicking on the "x" next to the label, also apply here. This option would minimize the breadcrumb but would not provide an immediate visualization of the tags affecting the filter. Nevertheless, we have to remember that all the negated tags will be always visible in the facet menu.

At any case, what seems obvious is that the option to delete all the negated tags from the filter in one click is vital. Therefore, we feel the need to comment that in the case the "facet names" were introduced, understanding such names would be unique, a completely negated facet could be represented in the breadcrumb with the "NOT facetName", minimizing the current approach. However, this improvement would not be of much use if individually negated tags within a facet were introduced.

Facet Menu: None of these																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
Users remove tags from facets thinking they are removing the restriction	X	-	-	-	X	X	X	X	-	X	X	X	-	-	X	X
The "none of these" tool enhances single-valued facets	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Showing tag negations with neither/nor is visually ugly	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 6.7.: Usability issues from the "none of these" options found in the facet menu.

6.1.3. Add to/remove from all

The add to/remove from all feature is placed in an appropriate location because users immediately understand that the tool only affects the list of resources. Additionally, it is enough with defining a default set value with ”add/remove tag to all” for users to understand what the tool is for. Tooltips for the specific ”+” and ”x” buttons might be helpful even though none of the testers has needed extra help to understand what they meant. A discussion might arise from the decision of using ”x” instead of ”-”. Depending on the point of view, ”-” might be found more suitable, however, ”x” is used to be consistent with the ”x” users needs to click on for manual individual deletion of tags from resources. If ”x” means the removal of a tag in a feature, it has to mean such thing in all the features of the system.

Probably the main issue involving this feature is that of the required time to apply its changes, especially for large result sets. This fact has been notwithstanding greatly improved by the implementation of the rules in version 4. However, it would be advisable to provide feedback to the users on how many resources have been affected, because not always all the resources of the result set are affected. It could even be possible that none was affected, when adding a tag in the case where all the resources already have such tag or when removing a tag where none of the resources has it. Providing feedback also ensures to the users that something has happened.

A further problem is that when tags involving subsumption relations are added using these bulk options, the related tags are not automatically added, thus breaking such relations. However, if subsuming tags were automatically added to the resources when using a bulk operation, unapplying tags would not be easy to find nor remove. Nevertheless, for an specific filtered context, deduced tags should generally be correct.

Providing the option to drag tags into the input might also speed up the management of resources and tags within TACKO.

Finally, it might be interesting to consider the pool of tags from which the autocomplete should provide its suggestions. In the case of the ”remove” option it seems obvious that the autocomplete should only offer tags that are currently attached to at least one of the resources of the current set. However, for the ”add” option almost any tag can apply. Our tests don’t provide enough information as to get to a conclusion on if the tags normally added using this tool are: tags already in the current context, so that they are added to heterogenize the current result set, tags already in the system but not related to any of the resources of the current set, which is quite unlikely, or tags that are totally new to the system, which is quite probable. From these observations we would conclude that probably the best option is for the autocomplete to suggest only tags that have matches in the current result set because this way users would not be suggested with tags that cannot be removed from any resource.

Add to/Remove from all																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
No feedback on affected resources is provided	-	-	X	X	-	X	-	-	-	-	-	-	X	-	-	X
Bulk operations do not support subsumption relations, generally breaking them	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 6.8.: Usability issues from the add to and remove from all feature.

6.1.4. Resource List

Several actions in the interface involve changes into the list of resources, and when some of this actions require more than 2 seconds to refresh feedback should be provided to the users. In version 4, though, after the implementation of the rules, no *spinners* are needed for this feature. However, it is important that the lists are consistent, for example, for a certain filter, the list of resources should always return the same output. In other words, for a filter formed by tags A, B and C, independently of which of the 6 possible combinations to choose these tags, the resulting list of resources should be the same, in the manner that the items are always shown in the same order. In case of changes that affect the result set of a certain context, then these changes have to be shown with the minimum implications, i.e when a resource doesn't comply the current filter anymore, then the following resources should move one position upwards. This is necessary because users tend to get familiar with the output of the main part of the interface, the resource list, and when unexpected changes appear here, they might get confused. Showing features like the resource counter helps users to get their bearings more easily. Currently, the system behaves in such manner in most of the cases but there are few exceptions. For example, sometimes for a certain context, if an operation that doesn't modify the state of the tags of the system is applied, such as going back and forth, checking a deeper level, or clicking the none of these label, when returning to such context the first resource of the list disappears from the first position. It is still in the list but it gets lost among the other resources in a random position. This is theoretically not a problem, nor wrong, but confuses the users, because when they go back to such context they expect to find exactly the same list that was there when they left. This little alteration affects the capability for users to *know where they are*. Especially, in visual elements like with the list of flickr photos this becomes a problem, because users tend to involuntarily associate contexts with the first picture they see, and if this first picture changes, then they initially think they are on a different context. This makes the system appear incoherent. There is no specific scenario where this problem occurs, just after every several clicks or certain (unknown) combination of clicks the first item of the list will disappear from that position. It is a very hard to reproduce behaviour.

For lists of flickr photos, after clicking on the preview of a picture, it is not obvious that to close the image you have to click on the image again (f.e in Facebook clicking on the image will mean next image). Users tend to go to the "x" at the top left corner only after realizing that clicking out of the picture does not do the work. All the users try to click out of the image because this is the behaviour of most of the websites. However, it would interesting to check how usual users from Flickr, where the same behaviour is used, act in this situation.

Even though the need to click on a certain button to access the edit mode of a resource slows the process of categorization, it has not been found as a relevant issue. However, alternatives could be offered for those cases when users decide that adding certain tags manually to different resources is faster than applying a filter. Normally, this happens when users find 2 or 3 images in a row that are missing a tag. Applying a filter doesn't make much sense in this scenario and users prefer to edit these resources one by one. For this occasion or others where it's simply impossible to find a filter that will return the exact result set of resources, the following possible improvement is suggested. It basically consists of permitting the users to select resources from the result set to which they want to apply changes. The current tools are very powerful for already tagged datasets which already form a significant *folksonomy*, because they permit to work in relation to other tags of the system. However, for datasets where the *folksonomy* has to be created from scratch, the tools offer certain limitations. Imagine the scenario where no tags (and therefore no facets) are yet introduced to the resources of the system. Initially, filtering options won't apply. Either you add the same tag to all the resources, or you add distinct tags to resources one by one. A relevant problem appears when the same tag has to be added to several resources, and this group of resources cannot be filtered yet. In this case, a user would need to type the same tag in every resource edit mode, or if smarter, will add it to a one-tag facet to drag it to every resource to where it applies. However, dragging a tag to let's say more than 10 resources is a tedious work. This problem does not necessarily appear only in scenarios with *still-to-be-built-folksonomies*, but also in already defined *folksonomies* where new dimensions of categorization might want to be defined. In other words, if in our set of flickr photos we had not initially defined tags regarding meteorological conditions, and we wanted to add them now, how could we filter the set in a smart way so that we can add *rain* to several pictures at once? Maybe, filtering by *bergen*, where it rains 300 days a year would be quite accurate, but this is of course not an acceptable solution. The case is that in TACKO, new tags that offer new dimensions of categorization that cannot be yet found in the dataset have more limited management options. Offering users the possibility to select resources from the list, i.e marking them in light gray once they click on the resource's area, would permit them to add a certain tag to a group of several resources that could not be filtered. For example, the "add to/remove from all" feature could be used as the input for such tool. This way, users could browse through a list of pictures selecting all of those where rain manifests to finally type *rain* into the input and add it to all of them. This tool can also be very useful in cases where a tag has to be removed from several resources that cannot be filtered with the current tags of the system, but not from all of the ones containing it. Additionally, a certain unfilterable subset of resources might need a tag to be renamed. This tool would also apply here.

Of course, such tool would require some interface changes but these can be considered minor. As we have already introduced, the input for bulk operations could be used in this case as well. Additionally, it must be said that using such input for this purpose would make sense, as the use we give it in the current version can be understood as an specific case of the use we now want to give to it. An specific area of every resource should be specified to make it "selectable", so that it doesn't conflict with the edit mode. In the specific case of bulk renaming, it would also be interesting to modify the "add to/remove from all" feature so that it supports renaming. Renaming selected resources

would only rename the tags of those selected resources, while renaming a tag in a certain context would only rename it in the resources of that context, something that cannot be done at the moment. The usefulness of this feature seems obvious, homonymous tags could be easily distinguished, i.e with the use of synonyms, with this tool because filtering options help to classify their meaning differences.

We feel the need to comment that, despite it is true that such operations would have no direct relation to TACKO, they would provide better means for users to manage their pool of tags, thus improving the quality of tags, fact that would have a positive impact on TACKO and its singularities.

We already commented the problem that appears with the facet menu, the breadcrumb and the bulk operations tool disappearing from the viewport when users scroll down. With the implementation of a "show more resources" that substituted the current pagination this problem would be emphasized. In this direction we propose that either these features follow the user while scrolling down, or rephrasing it, making this features fixed in the interface, so that scrolling down only affects the resource list. There are different ways to solve this problem but what seems obvious is that such features need to be always visible.

Resource List																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
First element of lists randomly disappearing	X	-	-	-	-	X	-	-	-	X	-	-	-	X	-	X
Need to click inside to close a picture	-	X	X	-	X	X	X	-	X	X	-	X	X	X	X	X
Impossibility to manage a tag for several unfilterable resources	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Impossibility to rename a tag for a certain context	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 6.9.: Resource List usability issues.

6.1.4.1. On edit mode

Probably the most relevant issue of the "edit mode" for a resource in the context of TACKO appears at the time to add a tag that defines *subsumption* relations, so that other tags are automatically added. Normally, in this situation, users realize new tags have been added automatically but this is not always the case. Especially in the rare situation where the automatically added tags do not fit in the same line and are added into a second row of tags. They might realize in future apparitions of similar situations, in the way that at some point they learn this might happen, but for then it might already be too late for previous subsumption relations that did not apply. Therefore, more eye-catching indicators might be useful. However, it must be said that this is not only a rare issue but also with relatively insignificant consequences.

Users generally understand the underlying idea of why those tags were automatically added, mostly because of "is-a" or "part-of" relations, though they do not generally do this reasoning. The problem here arises in the fact that they don't know how those relations were defined, nor if they are computer-generated or extracted from any of their

actions, which is the case. Therefore, they just see it as a useful feature that works in parallel and independently from them, without considering if they can affect its behaviour. Obviously, they have no clue on facts such as removing an automatically added tag undoes the relation. We consider it would be interesting to somehow let the users learn how it works and how they can benefit from them, because understanding such feature could greatly enhance their use of the system. How to teach this only with the use of the interface raises as a challenging problem. Maybe the most feasible solution would be to introduce the concept externally from the system, with short example clips, etc.

In the case of automatically added tags because of subsumption, removing the tag that was added and made other tags to appear should make the whole set of new tags to be removed, at least if its done before saving the first time. The reason of this is quite obvious, for example if we say that *panorama* subsumes *june*, adding the latter to a resource will automatically add the former to the resource, and, if after adding a the tag *june* the user realizes that the resource was actually from *july*, then he will proceed to remove *june*. Here, two possible problems appear. The first is that the user might forget to also remove *panorama*, thus leaving an incorrect tag in the resource. The second is that, independently of if he remembers to also remove *panorama*, the action of removing those tags separately might have undone the relation of subsumption between them. Thus, we suggest that removing *june* also implies the automatic removal of *panorama*.

Faceted suggestions are clearly a powerful tool that can be only understood by users who have a good understanding of the TACKO system. For beginners, such tool is overwhelming and a cause of frustration. To begin with, the suggestions are based upon information previously provided by the users in the form of facets, so no real overview on tags unknown to the user is really provided. However, it is true that users had to perform alternatives that involved several moves, even among different contexts, in order to add a tag to a resource that could have been easily done if understanding how this feature works. Users immediately see the relation with the columns offered here and the facets of the current context but minimal alterations crush their reasoning. Some bugs that were propagated until version 3, presented in previous versions, could be observed through this feature but made the use and evaluation of this feature quite impossible. However, from tests in the latter version, it could be deducted that the problem with faceted suggestions arises from the inability of the feature to show the relations between facets from more specific contexts and the tags that had made them appear. Also, the order in which columns are shown or new columns appear is not obvious to the user. For example, using different colours or sizes that indicated different levels of the *hierarchy* or a representation that could express which combinations of the currently selected tags makes a facet appear would be a great enhancement for this tool. Nonetheless, this is not a trivial problem, and maybe teaching the users is more factible.

Additionally, the use of horitzontal scrolls should be avoided because they are generally disliked, and also complicate the possibility to get a complete overview of the represented features that need of such tool. For example, the understanding of faceted suggestions is impeded by the impossibility to view the overall set of columns.

A very insignificant issue is that users who enter the edit mode in resources from contexts without defined facets get no suggestions under the message that offers them. However,

this is not a normal scenario.

Finally, the fact that tags are represented inside the box representing the input and they have way greater visibility than the text cursor, relegates the latter to a secondary position in terms of identification and recognition. Considering it is the main action of the feature, its visibility should be enhanced in relation to its relevance.

Resource List: On edit mode																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
Users do not always realize of the tags added due to subsumption relations	-	X	X	-	-	X	-	X	-	X	-	X	-	-	-	-
Users are unaware their actions affect the deduced tags from subsumption relations	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Automatically added tags because of subsumption should also be automatically removed when the subsumed tag is removed	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Faceted suggestions are not representing their utility in a proper way	X	X	X	X	X	X	X	X	X	X	X	X	-	X	-	X
When no facets are defined, no suggestions are offered	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
The text cursor to add new tags is not noticeable enough	X	X	-	X	-	-	X	-	-	-	X	-	-	X	-	-

Table 6.10.: Resource List usability issues when on edit mode.

6.1.5. Others

One of the most commented issues users have commented is that the cryptic paths the system generates do not help them get their bearings nor immediately access a certain context. The paths of the application are simply unbearable and almost all of the tested users dedicated few words to it without leading them to this observation. Lately, the paths have become very important, both for users to orient themselves within the system and to define restrictions. For example, in Delicious.com searches by several tags can be performed by separing tags in the path with "+". In reddit.com several *subreddits* (custom-made subforums) can be aggregated together simply by separing subreddit names with a "+" in the path, forming the so called *multireddits*. While Delicious.com offers the possibility to create such filters through the interface, reddit.com does not, so it really works as a tool used by the most advanced users who necessarily need to type them in the path. Offering such possibilities with TACKO would be very useful. At least, paths should be made readable.

As TACKO allows tags and negated tags as filters, a more complex path is required to be built. However, something of the nature of /wikiName/resourceType/tag+tag+tag+tag/negatedtag+negatedtag (where / acts as a separator between tags and negated tags) or something more representative such as /wikiName/resourceType/tag+tag+tag+tag-tag-tag-tag should be considered. "/" are generally related to hierarchies so its appropriateness to define hierarchies in TACKO should also be contemplated.

Confirmation messages, as the ones that appeared in version 1, are generally disliked and seen as a slowing step. Users believe they take responsibilities by doing their actions and they do not want to confirm them. Of course, they are wrong with this behaviour but they only learn by making mistakes.

Finally, a possibly interesting feature would be that tags can be renamed without needing to add them into a facet. For example, doubleclicking any tag in the system could offer renaming.

Others																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
System paths are unbearable	X	X	X	-	X	X	-	X	-	X	X	X	X	X	X	X
Confirmation messages are annoying	X	X	-	X	-	-	-	-	-	-	-	-	-	-	-	-

Table 6.11.: Other usability issues.

6.2. Observations derived from the tasks

In this section we will briefly comment other observations derived from the tasks that do not necessarily involve usability issues, but that might be interesting to consider at the time to apply changes to the interface.

At the time to search for a certain picture, users tend to follow two different strategies. Some normally start filtering by the most specific information they find on pictures with a bigger risk for the resource to fall out of the result set, expecting to quickly narrow down the result set. This approach either helps them find the resource especially quickly or forces them to undo several false steps. On the other hand, others start adding tags that narrow down the result set more slowly but making more secure steps, because they use tags they are quite sure the resource will contain. We could say that the first group of users apply a trial an error process, while the second group of users tend to analyze in more detail every situation.

Users are so used to search by typing tags or text that they initially ignore every other search tools. For example, in this work, the use of the breadcrumbs had to be forbidden for some tasks so that users would need to use the facet menu. Generally, users prefer to use the *typing* search options to find resources they know exist but they have not categorized. Alternatively, users show no preferences between search options, and tend to use both the facet menu and the input of the breadcrumb at the time to find resources categorized or tagged by them. In other words, when they know *the way to go*.

For organization tasks, several different approaches have been observed, which could have some affectation on the creation of facets when using the system collaboratively because these different styles of categorization do not necessarily need to work together. For example, different users would define a facet representing the same concept, i.e with the same tags, in different contexts of the system. This behaviour can be generally divided into two groups of users, those who always try to define facets in the most general contexts, thus

being able to dispose of them anywhere in the system where they might apply, or those who always try to define them in the most specific contexts possible, where they surely are used and relevant. The behaviour of the former group tends to overcrowd the interface of facets, some of which might be irrelevant in the current context, though users seem not to dislike this. However, they tend to ask for options that would let them reorder facets within a context, so that they can order them by relevance, etcetera. The latter group, on the other hand, tends to have few facets per level, but in some cases this might require them to apply more steps to get to the desired result set. It is not yet clear which is the best approach, and of course users do not strictly fall into any of both groups but tend to behave more in one or the other directions.

Additionally, for early versions where deleting facets in a context did not affect more general contexts, some users would tend to define all those facets they could come up with in the empty filter context, even if the facet's tags were assigned to less than a 10% of the resources of the context. This way, they had all the dimensions of categorization in all the contexts of the system. In addition, as in the empty filter context all the tags of the system had matching results, they could be always seen in the interface (at that time, tags that did not appear in any of the resources of the current context were not shown). This means that facets defined in the empty filter context provided a feeling of completeness they could not achieve when defining such facets in more specific contexts were some tags might be hidden. Moreover, the fact that while accessing more specific contexts, unmatching tags and facets without matching tags were not shown anymore, enhanced this behaviour. These facts also helped users to realize that they were successfully narrowing down the search when searching for a resource, because every time less possibilities were offered to them, thus requiring less challenging cognitive efforts to keep on narrowing down the search. We found this behaviour quite smart, though it did not make use of hierarchical concepts and that might become very unpractical for bigger sets of resources where the number of facets might increase notoriously. However, it is important to note that users were not worried with having several facets in the more general contexts because they normally tended to refine the resources categorization while on more specific contexts, where the number of facets was notably smaller.

Similarly, other users would start by defining all the facets in the empty context filter, except those that were of the "is-a", "part-of" nature, that they decided to define inside their respective according context filters. We could understand this as a refinement of the previous approach, but very irrelevant facets were still defined in general contexts. Additionally, some users would decide to delete these irrelevant facets but only in more specific contexts where they proved to be useless. This led to curious situations. For example, a facet formed by *ferry* and *kayak* (with 5 and 2 hits respectively) could be created in the empty filter context, to be deleted in all the one-tag filters were it was shown (where it was not shown because of no matching hits, it could not be deleted). It seems like some users wanted to have a general overview of the dimensions of categorization in the empty filter or "home", even if they were irrelevant in more specific contexts. This also helps to think that users tend to distinguish the empty filter, from all the others, at a bigger scale that one would expect. This makes sense if we understand users do not think of a filter as "empty", but understand it as a pool of resources from where some are filtered out.

To exemplify the previous approach we could say that users tended to define facets representing countries, years, transport types, meteorological conditions, building types or geographical features in the first level, while for example defining "cities of norway" inside the context filter of *norway*. If there were mostly pictures of several transport types inside norway, then they deleted the facet from the other contexts where few hits resulted while not being able to remove them from contexts without hits. This resulted in a final inconsistent structure which did not help at all. Because of such behaviours among others previously presented we would find very interesting the introduction of the forementioned *facet header*. For a better learnability, these should be shown expanded, so that the facet menu does not aggressively diverge among contexts, where only unmatched tags might change in colour. For a better efficiency, these should always appear collapsed, so that facets without matching results did not bother, but were still shown so users would know of their existence in the context.

Another interesting observation is that users would create a facet selecting all the suggested tags, thus creating a big group of tags representing no clear concept, so that afterwards they could categorize them into other facets by simply dragging them. These made the process more efficient because users only needed to go through every tag once, at the time to decide to which facet it applied. This is a quite interesting backing fact for the introduction of a "rest" label that showed all the unattached tags to any facets for a certain context.

Similarly to the previous behaviour, some users would define a facet for all those tags whose meaning they did not know, in other words, tags they did not know how to categorize, but that they somehow knew they had to categorize because they had a large amount of hits in the result set. This way, they improved the suggestions for facets whose concept they clearly knew. Finally, in some cases, they would filter using those unknown tags to get a better idea of which concept each might represent, to go back and place them in their right facet.

Other certain users, maybe with a better knowledge of the set of tags and what they represented, opted for defining very obvious facets in the first level, such as countries, years and months, and then they would drop other tags (previously haphazardly added to a facet without concept) into those concepts they related them to, like *fjord* inside *norway* or *stockholm* inside *sweden*. This behaviour frequently introduced wrong subsumption relations, for example *ferry* into *norway*, but these could be refined later. Additionally, when accessing a filter context, i.e *norway*, they would merge the several newly created facets into their respective concepts.

In general, it can be said that users start categorizing those concepts they are more familiar with. Also, it has been observed that users tend to create single-valued facets. Some users even doubt about defining tags that can appear together in resources as concepts of the same facet.

Finally, still in the context of organization tasks, it can be observed that different types of users exist depending on their level of thoroughness at the time to categorize the collection. Some prefer to organize in detail and can get frustrated if they don't immediately understand the behaviour of the system or master its multiple tools, while others do not give so

much importance to it while they are still able to find relatively soon what they need. This latter group is normally not even considering if the system helps them retrieving resources more easily that other traditional systems.

For the discovery tasks described in the previous chapter, where users were not concretely searching for something, these would not use the breadcrumb input anymore but their moves would be constrained by the navigation options provided by both the facet menu and the resources themselves. Generally, users would choose certain tags that drew their interest getting into filters of about 3 or 4 tags and almost never stepping back for the first two of the chosen tags. The system provides an interesting way to navigate into contexts that would have been unnoticed otherwise, thanks to the knowledge of other users embodied at the time to categorize tags into facets.

Regarding the differences among users from different groups, it appears to be quite obvious that their behaviours are quite diverging, especially in the context of organization tasks, where participants from group 1 would be almost unable to make meaningful use of the facet menu, participants of the group 2 mimicing the predefined facets but in inappropriate contexts and without having a clear idea of their aim, participants of the group 3, with an idea of their purpose but defining limited facets, like one-tagged or single-valued facets also in inappropriate contexts, and participants from group 4, who would try to customize the predefined sets into their own understanding and relevance of the concepts.

What is true is that some features are generally understood independently of the group, such as the breadcrumb and the bulk operations input.

6.3. Summary of the usability problems

Features																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
Breadcrumb																
Uses input as a search bar	-	X	X	-	X	X	-	-	-	X	-	-	-	-	-	-
Need to focus new input when manually adding tags	X	X	-	X	X	X	-	-	X	X	X	X	X	X	X	X
Autocomplete leads users to a cul-de-sac	X	-	-	X	X	X	-	-	X	X	X	-	X	X	X	X
Facet Menu: Interface design and behaviour																
System aborts tag selection	X	X	X	-	X	-	-	X	X	X	X	X	X	-	X	X
No initial facets	X	X	X	X	X	-	X	X	X	X	X	X	-	X	-	X
Groups of facets are not obvious enough	-	X	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Unnoticed add tag to create new facets	-	X	X	X	-	-	-	X	X	-	X	-	-	-	-	-
Adding a tag without matches has no effect (v3)	X	X	X	-	X	X	X	-	X	X	X	-	-	-	-	-
Tags without matches are not shown (v3)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Same tag could be added twice to a facet (v2)	X	X	X	-	-	X	-	-	X	X	X	-	X	X	X	-
Tags without matches could not be removed from facets (v3)	X	X	X	X	X	X	X	X	X	X	X	X	-	X	X	-
Tags without matches cannot be read when hovered over	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Facets getting added in different positions	X	-	-	-	X	-	-	X	X	X	-	-	X	X	-	X
Facets need to be removed tag by tag	X	X	X	X	-	-	-	-	X	X	-	X	-	-	-	X
Tags have to be added one by one (v2)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Dragging tags to last resources of the result set is inconvenient	-	X	-	-	X	-	X	-	-	X	-	X	-	-	X	X
Facet Menu: Structure																
Changes in facets do not affect more general contexts	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
List of selected is confused with other facets (v1)	-	X	-	X	-	X	-	-	-	-	X	-	X	-	-	-
Tags are ordered alphabetically within facets	-	-	X	X	-	-	-	X	-	-	-	-	-	-	X	-
Creating facets with unmatched tags made invisible one-tag facets (v3)	X	X	X	-	X	X	-	X	-	X	-	X	X	X	X	-
The same tag could be added several times to the same facet	X	X	X	X	-	X	-	X	X	X	-	X	X	X	X	X

When a tag is added to several facets, it cannot be removed uniquely from one	-	X	-	-	-	-	-	-	-	X	-	X	-	X	X	X
Facet Menu: Suggestions																
Need to go through all the suggested tags every time users are adding a tag	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Need to click more suggestions every time a tag is added	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Facet Menu: Drag & drop																
No way to know tags are draggable	X	X	X	X	-	X	X	X	-	X	-	X	-	X	-	X
Drag & drop indicators are totally overlooked	X	X	X	X	X	X	X	X	-	X	X	X	X	-	X	X
Drag & drop can mean many different actions	-	-	X	X	X	-	-	-	-	X	X	X	X	X	X	X
Drag & drop a tag into another tag creates a hierarchical relation as well as a *subsumption* relation	X	X	X	X	X	X	X	X	-	-	X	-	-	-	-	-
Facet Menu: Right button																
No way to know these feature is implemented	X	X	X	X	X	X	X	X	-	-	-	X	-	X	-	-
After refreshing a renaming the tag shows as unrelated	-	-	-	X	-	-	X	X	X	-	-	X	-	X	-	X
Rename's dialog box requires to be confirmed with the mouse	-	-	-	X	-	-	X	X	X	-	-	X	-	X	-	X
Facet Menu: None of these																
Users remove tags from facets thinking they are removing the restriction	X	-	-	-	X	X	X	X	-	X	X	X	-	-	X	X
The "none of these" tool enhances single-valued facets	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Showing tag negations with neither/nor is visually ugly	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Add to/Remove from all																
No feedback on affected resources is provided	-	-	X	X	-	X	-	-	-	-	-	-	X	-	-	X
Bulk operations do not support subsumption relations, generally breaking them	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Resource List																
First element of lists randomly disappearing	X	-	-	-	-	X	-	-	-	X	-	-	-	X	-	X

Need to click inside to close a picture	-	X	X	-	X	X	X	-	X	X	-	X	X	X	X	X
Impossibility to manage a tag for several unfilterable resources	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Impossibility to rename a tag for a certain context	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Resource List: On edit mode																
Users do not always realize of the tags added due to subsumption relations	-	X	X	-	-	X	-	X	-	X	-	X	-	-	-	-
Users are unaware their actions affect the deduced tags from subsumption relations	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Automatically added tags because of subsumption should also be automatically removed when the subsumed tag is removed	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Faceted suggestions are not representing their utility in a proper way	X	X	X	X	X	X	X	X	X	X	X	X	-	X	-	X
When no facets are defined, no suggestions are offered	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
The text cursor to add new tags is not noticeable enough	X	X	-	X	-	-	X	-	-	-	X	-	-	X	-	-
Others																
System paths are unbearable	X	X	X	-	X	X	-	X	-	X	X	X	X	X	X	X
Confirmation messages are annoying	X	X	-	X	-	-	-	-	-	-	-	-	-	-	-	-

Table 6.12.: Summary of the found usability issues

6.4. List of main issues

Following, we provide a list of the main usability issues still left in version 4 in a decreasing order of importance:

Issue	Feature	Description
No initial facets	Facet Menu	Users have no examples in which to base their proper use of the system.
Facets getting added in different positions	Facet Menu	Adding a new facet does not always put the new facet on the same position within the facet menu. Complicates the understanding of facets.
No way to know tags are draggable	Drag&Drop	The system does not show tags are draggable in any way for users to realize.
Drag & drop can mean many different actions	Drag&Drop	Dropping a tag can mean ″copy into″, ″move″ or ″move into″ depending on where it is dropped.
No way to easily undo some operations	Other	For example, when dragging a tag on another tag, a hierarchy and a subsumption relation are created, not being obvious for the user to undo what he has done.
Impossibility to manage a tag for several unfilterable resources	Resource List	A tag might need to be added to several unfilterable set of resources. Right now it has to be done manually.
First element of lists randomly disappearing	Resource List	Sometimes the first resource of a result set disappears from the first position disorienting the users.
Drag & drop a tag into another tag creates a hierarchical relation as well as a *subsumption* relation	Drag&Drop	The system offers no way for users to know this.
After refreshing a renaming, the tag shows as unrelated	Right button	Users think they have done something wrong because the renamed tag appears with 0 matching results.
Users remove tags from facets thinking they are removing the restriction	None of these	Thinking they are simplifying the restriction of a set of negated tags, the users are actually removing tags from a facet.
No way to know right button options are implemented	Right button	The system offers no way for users to know this.
Need to go through all the suggested tags every time users are adding a tag	Facet Menu	Considerably slows the process and users search for alternatives.
System aborts tag selection	Facet Menu	Sometimes clicking on a tag has no effect. The browser starts loading a new context but suddenly aborts.

Dragging tags to last resources of the result set is inconvenient	Drag&Drop	Facets might fall out of the viewpoint when scrolling down the set of resources.
Automatically added tags because of subsumption should also be automatically removed when the subsumed tag is removed	On edit mode	Otherwise, in the cases of mistakenly added resources, the subsumption relation is involuntarily broken.
Faceted suggestions are not representing their utility in a proper way	On edit mode	Requires a deep understanding of how the system works in order to deduce its utility.
System URLs are unbearable	Other	System URLs do not help users getting oriented within the system, so that they do not see the possibility to reuse them.
Drag & drop indicators are totally overlooked	Drag&Drop	Noone realizes indicators on where to drop tags.
Bulk operations do not support subsumption relations, generally breaking them	Add/Remove all	Breaks previously defined relations.
Users do not always realize of the tags added due to subsumption relations	On edit mode	This might involve inconsistencies and several involuntary changes that might affect the quality of the subsumption relations.
Users are unaware their actions affect the deduced tags from subsumption relations	Other	They have no way to benefit from them without knowing how to define and undefine them.
No feedback on affected resources is provided	Add/Remove all	It is not obvious to see if the feature has worked, no feedback is given on its success.
Impossibility to rename a tag for a certain context	Other	If a tag renaming only makes sense for a certain context, it cannot be done. The tag must be deleted and added again using bulk operations.
Need to click more suggestions every time a tag is added	Facet Menu	The system does not remember users have chosen ”more suggestions”, thus slowing the process of defining facets.
Tags without matches cannot be read when hovered over	Facet Menu	The light blue colour in which unmatched tags are represented is not readable with grey highlighting.
Need to focus new input when manually adding tags	Breadcrumb	After manually adding a tag using the breadcrumb input, users need to focus the input again with the mouse for adding a further tag to the filter.
The ”none of these” tool enhances single-valued facets	None of these	Two tags of the same facet cannot be added to a resource with the use of this feature.
When a tag is added to several facets, it cannot be removed uniquely from one	Facet Menu	Gives the user the feeling that being able to add the same tag to several facets is a bug.
Showing tag negations with neither/nor is visually ugly	Breadcrumb	Sets of negated tags are represented with ”NEITHER tag NOR tag2 NOR tag3...” which is generally disliked.

Autocomplete leads users to a cul-de-sac	Breadcrumb	Autocomplete does not check if it makes sense to suggest certain tags in specific contexts, where they might lead to empty result sets.
The text cursor to add new tags is not noticeable enough	On edit mode	Users do no immediately realize where to add new tags because the text input is too hidden.
Facets need to be removed tag by tag	Facet Menu	There is no way to directly remove an entire facet.
Uses input as a search bar	Breadcrumb	The first time a user uses the breadcrumb's input, he expects to do a search.
Unnoticed add tag to create new facets	Facet Menu	The lone "add tag" input is not necessarily interpreted as a way to create a new facet.
Need to click inside to close a picture	Resource List	A photo requires a click inside or in the "x" at the top left, instead of clicking outside its are, to close it.
Tags are ordered alphabetically within facets	Facet Menu	Sometimes some other orders might make more sense.
When no facets are defined, no suggestions are offered	On edit mode	No tags are suggested under the message offering them in contexts where no facets are defined.
Groups of facets are not obvious enough	Facet Menu	Bigger separation between tags does not always lead users to observe the concepts the different groups represent.
Rename's dialog box requires to be confirmed with the mouse	Right button	Users expect that enter should enable them to confirm every element of the interface.

Table 6.13.: List of issues left in version 4 ordered by relevance.

6.5. List of possible improvements

Following, we present a summary of the possible improvements for the TACKO interface divided by the feature they would apply to:

Breadcrumb

- Input should state ”filter by tag” as default set value.
- Use autofocus on the input after having added a new tag, so that a new one can immediately be added without needing to focus the input with the mouse.
- Distinguish tags that are not assigned to any facet in the context of the filter defined by the set of previous tags of the breadcrumb.
- Add a dropdown option for each tag offering other tags from the same facet.
- Autocomplete should prioritize tags contained in any of the facets of the current context.
- Autocomplete should not show tags that do not have matching results in the current context.
- Add a dropdown option that englobes all the negated tags in the filter.
- Add an option to directly add individually negated tags.

Facet Menu

- Show unrelated tags in the facet menu. Done in version 4.
- Incorporate the option to collapse unrelated tags to shorten lists.
- The *facet header* defined in Section 6.1.2.1 involves several improvements.
- The facet menu should follow the user while scrolling down the result set.
- Changes in facets should affect these independently of their context. Done in version 3.
- Distinguish inherited facets from facets defined in the current context might prove valuable.
- Place inherited facets in the first positions and newly created facets at the bottom of the facet menu.
- Animatedly show facet menu changes when changing levels of specificity.
- Add a *tags in none of these* or *rest* feature that lists all the tags existent in the current result set that are not attached to any facet.
- Let users discard tags from the suggested list of tags.
- Show tags are draggable with the use of the *move* cursor.

- Redistribute the actions in the facet menu so that a right button menu is not required.
- Individual tag negations should be offered in the breadcrumb.
- The possibility to partially negate facets should be offered.

Add to/Remove from all

- Offer bulk operations for renaming tags in certain contexts as well.

Resource List

- Offer management options such as selecting items from the resource list to work with unfilterable subsets of resources.
- Better representation of the utility of faceted suggestions.

Other

- Better and clearer paths.
- An option to rename tags should be offered without needing to add them to a facet.
- Easy ways to undo advanced operations such as dragging a tag into another one, i.e rollback options, should be offered.

7. Summary & Outlook

In this work we have presented the usability evaluation of the interface of the multi-faceted, tag-based knowledge organization system TACKO. We have first provided the necessary background knowledge regarding the tagging systems in order to help the understanding of TACKO and its singularities. Additionally, we have presented a quite extense overview of the most common usability evaluation methods, both inspection and test methods, some of which we have used for the evaluation presented in this work.

The work itself has consisted of the testing of the interface with a total of 16 testers, whose backgrounds are strongly related to informatics, to evaluate how easy to learn is the interface and as to which extent does the interface help users understand the underlying advantages and benefits the TACKO system incorporates. Knowing which is the minimum amount of information the users need to know for a successful search, organization and discovery of the resources of a collection with TACKO was also found necessary. For the development of these tests, users have been grouped in four different groups that diverged in the amount of the previously given information they disposed from. These four groups of users have separately tested the different versions of the interface through a set of tasks we gave to them. The tasks focused on the different scopes we wanted to evaluate, such as search & retrieval of resources, organization of the resources with the use of tags, hierarchies and facets, and discovery of resources using the offered navigation options. These tasks had different levels of difficulty and their exhaustive application depended considerably on the testers availability and commitment.

For the proper evaluation of the system, we developed a simpler version of the current prototype that enabled users to import collections of tags and resources from both, the social bookmarking site Delicious.com and from the photography sharing site Flickr.com. However, most of the tests were conducted with sets of images imported from the latter, because tests with image collections have been found more valuable. This interface prototype especially developed for the testing was also implemented as an extension of the commercial web-based enterprise collaboration software Tricia.

One of the main goals of this work was to learn what did users really need to know in order to be able to start working with the interface with a certain level of understanding and mastery. Surprisingly, we have found out that the users can efficiently use the system without knowing as much as we expected of the underlying system design. However, some deficiencies on the interface have been found, that do not help users to understand certain concepts or that can mislead them to a wrong use of the system.

Users generally understand the idea of the existence of different contexts, even if they do not use such term. They understand these contexts depend uniquely on the set of tags of the filter, and they understand that the order in which these filter is created is irrelevant.

Users need to know that facets represent a concept, and thus, can be used to englobe tags that can be related to that concept. Additionally, users need to know that defined facets in one context will be inherited in more specific contexts, that is, contexts whose filter is an extension of the filter where the facet has been defined. Users generally understand the different levels of specificity of contexts, i.e if a context is more specific than another. Also, users see that facets can be created in every context, independently of its level of specificity. Additionally, users need to know that hierarchical relations can be defined among facets to ease the navigability of the system.

At this point, if users can comprehend this concepts, we can consider they have apprehended the basic necessary knowledge to use the system efficiently. However, for a better use of the system and a better exploitation of the TACKO's design singular properties, an effort has to be made to reduce the need of technical explanations. The interface should enhance these properties, so that their use can be learnt from it without users needing to be taught. One of the features that suffers more from these limitations are the subsumption relations, because users have no way to know if those are defined by them or the system, nor as to which extent does their behaviour affect them. This generally leads to a bad use, or more concretely, to a disuse. Other important limitations from the interface are the impossibility for users to know that draggable options are implemented, as well as right button options. Probably, the issue that involves more difficulties for the users at the time to understand the system is the inconsistency of the representation of facets among contexts. Nonetheless, it has to be said that the system has notoriously improved in this aspect thanks to the last upgrades. More work has to be done in the interface so that it supports and enhances the properties of TACKO.

During the explanation of the system and how it works to the different users, it is important to consider what terms do they need to know and which ones should be avoided. For example, during the execution of our tests we found very valuable that users learnt some new concepts basic for the understanding of TACKO, such as *facet* and *filter*. On the other hand, more advanced technical terms such as *single-valued* or *subsumption* should be avoided. For the former terms, it would be interesting to introduce them within the system, so that users learn them and their use becomes part of their vocabulary the same way *tag* and *folksonomy* is part of ours now. This is found valuable because we consider something cannot be understood until it can be explained, and users certainly need to know such concepts at the time to express themselves. More than half of the testers had serious problems to express what they had understood of the system, even if they affirmed they had understood it, while the majority made use of words such as "groups of tags of similar concepts" when referring to *facets* and "list of resources and groups of tags that depend on the set of tags in the breadcrumb" when referring to context, which made their explanations convoluted. For the latter terms, they should be hidden from users because they do not need to know them to efficiently define their concepts even though the interface should stress their existence within the interface. In the current versions, users are not aware of the possibility to define single-valued or multi-valued facets nor the implications these differences can have in their categorization, thus being impossible for them to even consider how to benefit from them. The same applies for the subsumption relations. If noone knows about their existence, noone can benefit from them.

The representation of some concepts in the interface may not be trivial and therefore, in some case it might be interesting to provide examples so that users can copy behaviours. In the last years, many new applications need to provide simple videos or walkthroughs for users to grasp the basic concepts and this should be considered here, until some of the interface features become more common. For example, not many users try to drag things nor search for right button options because these are still quite scarcely used options in the web. For example, smart tips that appeared at the top of the page depending on the users' actions would also be useful to unobfuscate the use of certain features.

Nevertheless, it ought to be said that the understanding of some features is independent to the previously provided information, while others are strongly dependant on it. For example, users mastered the breadcrumb without great difficulties when no information was given to them while many users had problems to understand the facet menu, even when they had been told how it worked.

What appears obvious is that the system's final categorization is hard to represent in one's mind and the fact that TACKO's benefits start to appear especially useful for big sets of resources, leads to the conclusion that only users who really need to work with large amounts of information will make the effort to understand it. Unless users really need its potential, it is hard to conceive that users would use it in substitution of other more traditional knowledge organization systems.

The system, as any other tagging system, strongly depends on the quality of tags and how well has a collection of resources been tagged. It is important to transmit this idea to the users, showing that more throughly tagged collections are of great benefit to them. It does not matter how well and efficiently we develop tagging systems if we fail to teach users good tagging practices, that impede the correct functioning of our implementations. Additionally, due to the big amount of different observed strategies applied by users, it appears relevant to also develop the interface so that users learn how should they use it and why. For example, one of the biggest concerns of users is that they were never sure of where to define a facet, because they were unable to imagine which implications defining it in one place or another would carry. The system should offer some guidance to mitigate this cognitive challenges.

It is from the varied approaches and strategies users follow at the time to use the system that groups of users can be observed. It still has to be observed how these different roles affect the system when being collaboratively used, but for now, we can see that while, some prefer to organize in detail and can get frustrated if they don't immediately understand the behaviour of the system, some others do not give so much importance to the categorization while they are still able to find resources relatively quickly.

The application of usability methods such as Constructive Interaction showed that the success of TACKO strongly depends on the ability users have to interpret organization structures made by others and to learn from more advanced users in order to benefit from the system themselves. However, it is important to note that in a collaborative system where the categorization is built by everyone users with different strategies might want to impose their own. Because of this and other social dynamics aspects, some control features should be implemented. These being in the form of access rights, so that not everyone

can modify the facets, or with other techniques such as version controls to easily undo changes as a less restrictive solution. Independently of the approach, it is important for the categorization not to be constantly aggressively changing because users familiarization with the categories would play no role at all.

Probably the main strength of the current interface is the simplicity in which facets are presented and efforts should be made so that this keeps being like this. Changes affecting its simplicity should be avoided as much as possible. The intercontextual consistency of the facet menu, that is, the similarity of the facet menu among the multiple contexts of the system is probably the most important attribute for the ease of users' understanding of the facets and their behaviour. Changes such as showing unmatching tags or facets of which some tags might have already been chosen can help to mitigate the variability of facets among contexts.

Additionally, enhancements similar to the proposed *facet header* could be introduced because they do not add much complexity to the menu while providing several advantages that boost the utilization of facets. Structuring the facet menu so that inherited facets are shown before facets defined in the current context would help the facet menu to comply with the consistency requirements.

Taking advantage of the more popular breadcrumbs, adapting them to our needs would also be an interesting solution. For example, showing the tags defined in the same facet for a certain tag of the filter in the form of a dropdown menu, would allow users to rapidly change the filter with concepts of the same dimension of categorization.

One of the most requested features has been the possibility to negate individual tags. The current version only enables to negate entire facets and therefore, if a single tag needs to be negated, a one-tag facet has to be defined. As its need becomes obvious for the management of the resources, we have proposed a possible way for the system's interface to adapt to such changes without affecting the simplicity of the menu.

Finally, the impossibility to efficiently manage tags for unfilterable subsets of resources, a problem that especially affects collections that are still not tagged, needs to be tackled. A possible solution that would enable users to select resources from the resource list to which an specific action would be performed is proposed.

We believe that as soon as some of the more more basic problems concerning the facet menu are fixed, the interface will be perfectly prepared for testing with real end users in the scope of a working prototype where the built *folksonomy* is product of the application of the system for real utilization.

Appendix

A. Users success with tasks

The following table represents the success of the diverse users with the various tasks.

Legend:
″s″ - successful
″sp″ - successful with slight problems
″u″ - unsuccessful
″-″ - not evaluated
″ne″ - not evaluable

Search																
Group.User	1.1	1.2	1.3	1.4	2.5	2.6	2.7	2.8	3.9	3.10	3.11	3.12	4.13	4.14	4.15	4.16
Lookup: Level 1	s	sp	s	s	sp	s	sp	s	sp	s	sp	s	s	sp	s	s
Lookup: Level 2	s	sp	-	s	s	s	sp	s	sp	-	s	s	s	sp	s	s
Lookup: Level 3	s	sp	-	sp	s	sp	-	s	sp	-	-	sp	s	sp	s	s
Described picture	s	u	-	-	s	u	-	-	u	-	-	sp	s	s	-	sp
Organize																
Single facet definition	sp	sp	u	u	s	s	sp	s	sp	s	sp	sp	s	s	s	s
Full dataset categorization	u	u	-	-	sp	u	-	u	sp	-	-	-	s	-	-	sp
Facets in the big dataset	u	u	-	-	u	u	-	u	u	-	-	-	sp	-	-	u
Discover																
Free exploration	sp	sp	-	-	s	sp	-	-	s	-	-	sp	s	-	-	sp
Favourite picture	ne	ne	-	-	ne	ne	-	-	ne	-	-	ne	ne	-	-	ne

Table A.1.: Users success with tasks

B. List of features

Header
tagImporter links to /importer
Breadcrumb
Home logo links to first level
Input for tags
Autocomplete with tags already in the system
If you click a tag you remove the following from the filter
If you click the X you delete that tag from the filter
NEITHER tag NOT tag2 for negated sets of tags
Facet Menu
Add tag adds a tag to facet or creates facet
List of tags are suggestions
”More suggestions” shows more suggestions
None of these
The number next to each tag is the number of resources containing such tag
Input is for tags
Autocomplete with tags already in the system
Unlimited facets per level
’Unlimited’ tags per facet
Tags might overlap
Clicking on a tag adds it to the filter
When hovering, if clicking ”x” deletes tag from facet
Drag a tag into another facet
Drag a tag into the resource
Right button rename
Right button delete
Right button quit
None of these
Left menu shows tags that are excluded
Add tag adds more excluded tags
The number next to each tag is the number of resources containing such tag
Clicking on an excluded tag returns to previous context
Add to/Remove from all
Input is for tag
”+” is to add to all resources in the list
”x” is to remove from all resources in the list

Resource List
Counter of resources for the current context
Clicking on the pictures makes them big
Clicking again makes them small
Click on the name links to resource in delicious or flickr
Click on a tag adds it to the filter
Click in the area next to tags or pencil → edit mode
Drag a tag into the facetMenu
On edit mode
Click on a tag → will remove it
Click ″x″ → will remove it
Save will save changes
Cancel will ignore changes
Tags in columns by facet
More suggestions in new columns
Grey means already attached to resource
Clicking on a grey tag → detaches tag
Clicking on a white tag → attaches tag
Click out of the edit mode will just cancel
Pagination
Clicking on a number will take you to that page
Clicking next will take you to the next page
Total result pages are shown

Table B.1.: List of features

List of Figures

List of Tables

Bibliography

[Bev95] Nigel Bevan. Usability is quality of use. *Proceedings of the 6th International Conference on Human Computer Interaction, Yokohama, Japan*, July 1995.

[Bia94] Randolph Bias. The pluralistic usability walkthrough: Coordinated empathies. *In Usability inspection methods. Nielsen, J. and Mack, R.L. New York, USA. Wiley*, pages 63–76, 1994.

[BKS06] G. Begelman, P. Keller, and F. Smadja. Automated tag clustering: Improving search and exploration in the tag space. *WWW 2006, Edinburgh, UK*, 2006.

[BPKL02] M. Blackmon, P. Polson, M. Kitajima, and C. Lewis. Cognitive walkthrough for the web. *Proceedings of the Conference on Human Factors in Computing Systems (CHI 2002), Minneapolis, MN, USA*, pages 463–470, April 2002.

[CR87] J.M. Carroll and M.B. Rosson. Paradox of the active user. interfacing thought: Cognitive aspects of the human-computer interaction. *The MIT Press. Cambridge, MA, USA*, pages 80–111, 1987.

[CSBT07] Y. Chen, R. Santamaría, A. Butz, and R. Therón. Tagclusters: Semantic aggregation of collaborative tags beyond tagclouds. *University of Munich and University of Salamanca*, 2007.

[DCSCS08] L. Di Caro, K. Selçuk Candan, and M.L. Sapino. Using tagflake for condensing navigable tag hierarchies from tag clouds. *KDD '08 Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1069–1072, 2008.

[DFAB98] A. Dix, J. Finlay, G. Abowd, and R. Beale. *Human-Computer Interaction, 2nd edn*. Prentice Hall, London, Great Britain, 1998.

[DR99] J.S. Dumas and J. Redish. *A practical guide to usability testing*. Intellect Books, Exeter, England. Portland, OR, USA., 1999.

[DS08] K. Dellschaft and S. Staab. An epistemic dynamic model for tagging systems. *In HT '08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia. New York, NY, USA.*, pages 71–80, 2008.

[EHS+01] J. English, M. Hearst, R. Sinha, K. Swearingen, and K. Yee. Hierarchical faceted metadata in site search interfaces. *Group of User Interface Research, University of California, Berkeley, CA, USA*, December 2001.

[Fag10] Jody C. Fagan. Usability studies of faceted browsing: A literature review. *Information Technology and Libraries*, pages 58–66, June 2010.

[Fau03] Laura Faulkner. Beyond the five-user assumption: Benefits of increased sample sizes in usability testing. *Behavior Research Methods, Instruments, and Computers 35 (3)*, pages 379–383, 2003.

[GB08] S. Greenberg and B. Buxton. Usability evaluation considered harmful (some of the time). *CHI 2008 Proceedings. Florence, Italy.*, 2008.

[GFA09] T. Grossman, G. Fitzmaurice, and R. Attar. A survey of software learnability: Metrics, methodologies and guidelines. *Boston, MA, USA*, April 2009.

[GH05] S.A. Golder and B.A. Huberman. The structure of collaborative tagging systems. *Information Dynamics Lab, HP Labs*, 2005.

[GLYH] M. Gupta, R. Li, Z. Yin, and J. Han. Survey on social tagging techniques. *SIGKDD Explorations*, 12, 1:58–72.

[GS98] W.D. Gray and M.C. Salzman. Damaged merchandise? a review of experiments that compare usability evaluation methods. *Human-Computer Interaction. George Mason University, VA, USA*, 13:203–261, 1998.

[Gun95] Cathy Gunn. An example of formal usability inspections in practice at Hewlett-Packard Company. *CHI '95 Proceedings*, 1995.

[Hea06a] Marti A. Hearst. Clustering versus faceted categories for information exploration. *Communications of the ACM*, 49, No. 4:59–61, April 2006.

[Hea06b] Marti A. Hearst. Design recommendations for hierarchical faceted search interfaces. *School of Information, University of California, Berkeley, CA, USA*, August 2006.

[Hea08] Marti A. Hearst. Uis for faceted navigation: Recent advances and remaining open problems. In *in the Workshop on Computer Interaction and Information Retrieval, HCIR 2008*, 2008.

[Hed08] Heather Hedden. Controlled vocabularies, thesauri, and taxonomies. *The Indexer*, 26, No. 1:33–34, March 2008.

[HGM06] P. Heymann and H. Garcia-Molina. Collaborative creation of communal hierarchical taxonomies in social tagging systems. *InfoLab Technical Report 2006-10*, April 2006.

[HGM09] P. Heymann and H. Garcia-Molina. Contrasting controlled vocabulary and tagging: Do experts choose the right names to label the wrong things? *WSDM '09, Barcelona, Catalonia, Spain.*, 26, No. 1:33–34, 2009.

[HHH08] Y. Huang, C. Hung, and J.Y. Hsu. You are what you tag. *Association for the Advancement of Artificial Intelligence . National Taiwan University, Taiwan.*, 2008.

[HJ03] M. Hertzum and N.E. Jacobsen. The evaluator effect: A chilling fact about usability evaluation methods. *International Journal of Human-Computer Interaction 15, Denmark*, pages 183–205, 2003.

[HK07] M. Halvey and M.T. Keane. An assessment of tag presentation techniques. *WWW 2007, Banff, Alberta, Canada*, pages 1313–1314, May 2007.

[HKG+12] D. Helic, C. Körner, M. Granitzer, M. Strohmaier, and C. Trattner. Navigational efficiency of broad vs. narrow folksonomies. *HT '12, Milwaukee, Wisconsin, USA.*, June 2012.

[HMHS06] Y. Hassan-Montero and V. Herrero-Solana. Improving tag-clouds as visual information retrieval interfaces. *In I International Conference on Multidisciplinary Information Sciences and Technologies, InSciT2006, Mérida, Spain*, October 2006.

[HN07] T. Hollingsed and D.G. Novick. Usability inspection methods after 15 years of research and practice. *Proceedings of the 25th annual ACM international conference on Design of communication, New York, NY, USA*, pages 249–255, 2007.

[Hol05] Andreas Holzinger. Usability engineering methods for software developers. *Communications of the ACM vol.48, No. 1*, pages 71–74, January 2005.

[HTSA] D. Helic, C. Trattner, M. Strohmaier, and K. Andrews. On the navigability of social tagging systems. *Knowledge Management Institute, Graz University of Technology, Graz, Austria*.

[HTSA11] D. Helic, C. Trattner, M. Strohmaier, and K. Andrews. Are tag clouds useful for navigation? a network-theoretic analysis. *Journal of Social Computing and CyberPhysical Systems (2011)*, 2011.

[Ins02] Keith Instone. Location, path & attribute breadcrumbs. *ASIST 3rd Annual IA Summit*, March 2002.

[ISO01] ISO. 9126-1:2001 software engineering – product quality – part 1: Quality model, 2001.

[ISO10] ISO. 9241-210:2010 ergonomics of human-system interaction – part 210: Human-centred design for interactive systems, 2010.

[JMWU91] R. Jeffries, J. Miller, C. Wharton, and K. Uyeda. User interface evaluation in the real world: a comparison of four techniques. *Proceeding of the Conference on Human Factors in Computing System (CHI 91), New Orleans, LA, USA.*, pages 119–124, April/May 1991.

[KCF92] C.M. Karat, R. Campbell, and T. Fiegel. Comparison of empirical testing and walkthrough methods in user interface evaluation. *Proceedings of the Conference on Human Factors in Computing Systems (CHI 92). Monterey, CA, USA.*, pages 397–404, May 1992.

[KLB04] J. Kontio, L. Lehtola, and J. Bragge. Using the focus group method in software engineering: Obtaining practitioner and user experiences. *Proceedings of the International Symposium on Empirical Software Engineering (ISESE)*, August 2004.

[KP94] M. Kahn and A. Prail. Formal usability inspections. *In Usability inspection methods. Nielsen, J. and Mack, R.L. New York, USA. Wiley*, pages 141–171, 1994.

[KR97] L. Kantner and S. Rosenbaum. Usability studies of www sites: Heuristic evaluation vs. laboratory testing. *ACM SIGDOC' 97, International Conference on Computer Documentation, Snowbird, USA (ACM, New York, USA).*, pages 153–160, 1997.

[KSHS09] B. Krause, C. Schmitz, A. Hotho, and G. Stumme. The anti-social tagger – detecting spam in social bookmarking systems. *AIRWeb '08, Beijing, China*, April 2009.

[KSS10] K. Knautz, S. Soubusta, and W.G. Stock. Tag clusters as information retrieval interfaces. *Proceedings of the 43rd Hawaii International Conference on System Sciences*, 2010.

[Lee06] Kathy J. Lee. What goes around comes around: An analysis of del.icio.us as social space. *CSCW '06, Banff, Alberta, Canada*, November 2006.

[LHY+09] D. Liu, X. Hua, L. Yang, M. Wang, and H. Zhang. Tag ranking. *International World Wide Web Conference Committee (IW3C2), Madrid, Spain.*, pages 351–360, April 2009.

[LJ06] K. Lerman and L.A. Jones. Social browsing on flickr. *Collaboration between USC Information Sciences Institute and Mills College, CA, USA.*, December 2006.

[LM03] J. Langford and D. McDonaugh. *Focus Groups: Supporting Effective Product Development*. Taylor and Francis, New York, NY, USA, 2003.

[LZT09] S. Lohmann, J. Ziegler, and L. Tetzlaff. Comparison of tag cloud layouts: Task-related performance and visual exploration. *T. Gross et al. (Eds.): INTERACT 2009, Part I, LNCS 5726*, pages 392–404, 2009.

[Mar06] Ivo Marinchev. Practical semantic web - tagging and tag clouds. *Cybernetics and Information Technologies, vol. 6, no. 3. Bulgarian Academy Of Sciences. Sofia, Bulgaria.*, pages 33–39, 2006.

[MC09] C. Mesnage and M. Carman. Tag navigation. *SoSEA '09, Amsterdam, The Netherlands.*, pages 29–32, August 2009.

[MN90] R. Molich and J. Nielsen. Improving a human-computer dialogue. *Communications of the ACM 33, 3*, pages 338–348, March 1990.

[MNBD06] C. Marlow, M. Naaman, D. Boyd, and M. Davis. Ht06, tagging paper, taxonomy, flickr, academic article, to read. *HT' 06, Odense, Denmark.*, pages 31–39, August 2006.

[MNS12a] F. Matthes, C. Neubert, and A. Steinhoff. Multi-faceted context-dependent knowledge organisation with tacko. *Technische Universität München, Munich, Germany*, 2012.

[MNS12b] F. Matthes, C. Neubert, and A. Steinhoff. Structuring folksonomies with implicit tag relations. *Technische Universität München, Munich, Germany*, 2012.

[MR92] B.A. Myers and M.B. Rosson. Survey on user interface programming. *ACM CHI'9 Conf.*, pages 195–202, May 1992.

[MRT] M. Matera, F. Rizzo, and G. Toffetti. Web usability: Principles and evaluation methods. Dipartimento di Elettronica e Informazione, Politecnico di Milano, Milan, Italy.

[Nei09] Theresa Neil. 6 tips for a great flex ux part 5. `http://designingwebinterfaces.com/6-tips-for-a-great-flex-ux-part-5` (Last accessed 03/08/2012), May 2009.

[Nie92] Jakob Nielsen. Finding usability problems through heuristic evaluation. *CHI '92*, May 1992.

[Nie93] Jakob Nielsen. *Usability Engineering*. Academic Press, 1993.

[Nie95] Jakob Nielsen. Usability inspection methods. *CHI' 95 Mosaic Of Creativity. Mountain View, CA, USA*, May 1995.

[Nie99] Jakob Nielsen. *Designing Web Usability*. New Riders Publishing, 1999.

[Nie06] Jakob Nielsen. Quantitative studies: How many users to test? `http://www.useit.com/alertbox/quantitative_testing.html` (Last accessed on 22/08/2012), 2006.

[Nie12] Jakob Nielsen. Thinking aloud: The #1 usability tool. `http://www.useit.com/alertbox/thinking-aloud-tests.html` (Last accessed on 06/08/2012), 2012.

[NM94] J. Nielsen and R.L. Mack. *Usability Inspection Methods*. Wiley, New York, USA, 1994.

[PFO+08] J. Park, T. Fukuhara, I. Ohmukai, H. Takeda, and S. Lee. Web content summarization using social bookmarks: A new approach for social summarization. *ACM. WIDM' 08, Napa Valley, CA, USA*, pages 103–110, October 2008.

[PRS+94] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey. *Human computer interaction*. Addison Wesley, Wokingham, UK, 1994.

[Qui05] Emanuele Quintarelli. Folksonomies: Power to the people. *ISKO Italy - UniMIB meeting, Milan, Italy*, June 2005.

[RD83] R.W. Root and S. Draper. Questionnaires as a software evaluation tool. *Proc. ACM CHI'83 Conf. Boston, MA, USA*, pages 83–87, December 1983.

[RGMM07] A.W. Rivadeneira, D.M. Gruen, M.J. Muller, and D.R. Millen. Getting our head in the clouds: Toward evaluation studies of tagclouds. *CHI 2007 Proceedings - Tags, Tagging and Notetaking, San Jose, CA, USA*, pages 995–998, April-May 2007.

[Rii] Sirpa Riihiaho. The pluralistic usability walk-through method. *Helsinki University of Technology, Finland.*

[RJ89] S. Ravden and G. Johnson. *Evaluating usability of human-computer interfaces: a practical method*. Ellis Horwood, Chichester, UK, 1989.

[SCDCS08] K. Selçuk Candan, L. Di Caro, and M.L. Sapino. Creating tag hierarchies for effective navigation in social media. *SSM '08, Napa Valley, California, USA*, pages 75–82, October 2008.

[Sch] Jean Scholtz. Usability evaluation. National Institute of Standards and Technology (NIST). Maryland, USA.

[SCH07] J. Sinclair and M. Cardew-Hall. The folksonomy tag cloud: When is it useful? *Journal Information of Science*, pages 1–18, 2007.

[SKK10] M. Strohmaier, C. Körner, and R. Kern. Why do users tag? detecting users' motivation for tagging in social tagging systems. *Journal of Emerging Technologies in Web Intelligence (JETWI)*, 2010.

[Smi08] Gene Smith. *Tagging - People-Powered Metadata for the Social Web*. New Riders, Berkeley, CA, USA, 2008.

[SP04] B. Shneiderman and C. Plaisant. *Designing the User Interface, 4th ed*. Addison-Wesley, Reading, MA, USA., 2004.

[SSR06] D.W. Stewart, P.N. Shamdasani, and D.W. Rook. *Focus groups: theory and practice*. SAGE Publications, London, UK, 2006.

[ST06] R. Santamaría and R. Therón. Overlaping clustered graphs: Co-autorship networks visualization. *University of Salamanca, Salamanca, Spain*, 2006.

[SvZ10] B. Sigurbjörnsson and R. van Zwol. Tagexplorer: Faceted browsing of flickr photos. *Yahoo! Labs Technical Report No. YL.-2010-005, Barcelona, Catalonia, Spain.*, pages 1–25, August 2010.

[SY99] N.A Stanton and M.S. Young. *A Guide to Methodology in Ergonomics: Designing for Human Use*. Taylor and Francis, London, UK, 1999.

[TLP+12] C. Trattner, Y. Lin, D. Parra, Z. Yue, W. Real, and P. Brusilovsky. Evaluating tag-based information access in image collections. *HT'12, Milwaukee, Wisconsin, USA*, pages 58–66, June 2012.

[Tog03] Bruce Tognazzini. First principles of interaction design. `http://www.asktog.com/basics/firstPrinciples.html` (Last accessed on 03/08/2012), 2003.

[TSHA] C. Trattner, M. Strohmaier, D. Helic, and K. Andrews. Potentials and limitations of tag clouds as a tool for social navigation. *KMI and IICM, Graz University of Technology, Graz, Austria*.

[TSMM08] J. Thom-Santelli, M.J. Muller, and D.R. Millen. Social tagging roles: Publishers, evangelists, leaders. *CHI 2008, ACM, Florence, Italy.*, 2008.

[vdHdJ03] M.J. van den Haak and M.D.T de Jong. Exploring two methods of usability testing: Concurrent versus retrospective think-aloud protocols. *University of Twente, The Netherlands*, 2003.

[Vir92] Robert A. Virzi. Refining the test phase of usability evaluation: How many subjects is enough? *Human Factors, 34(4)*, pages 457–468, 1992.

[VW05] Thomas Vander Wal. Folksonomy definition and wikipedia. `http://www.vanderwal.net/random/entrysel.php?blog=1750/` (Last accessed on 14/08/2012), 2005.

[War] Christian Wartena. Automatic classification of social tags. *Technical University Delft*.

[Wil11] Chauncey Wilson. Method 13 of 100: Consistency inspection. `http://dux.typepad.com/dux/2011/05/method-13-of-100-consistency-inspection.html` (Last accessed on 05/08/2012), 2011.

[WJTC94] D. Wixon, S. Jones, L. Tse, and G. Casaday. Inspections and design reviews: framework, history and reflection. *In Usability inspection methods. Nielsen, J. and Mack, R.L. New York, USA. Wiley*, pages 77–103, 1994.

[WRLP94] C. Wharton, J. Rieman, C. Lewis, and P. Polson. The cognitive walkthrough method: A practitioner's guide. *University of Colorado at Boulder*, 1994.

[WZB] R. Wetzker, C. Zimmerman, and C. Bauckhage. Analyzing social bookmarking systems: A del.icio.us cookbook. *Collaboration between Technische Universität Berlin, Cambridge University and Deutsche Telekom Laboratories*.

[YS05] M.S. Young and N.A. Stanton. Applying interviews to usability assessment. *Handbook of Human Factors and Ergonomics Methods*, pages 29.1–29.6, 2005.

[YSLH03] K. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. *CHI 2003, Fort Lauderdale, Florida, USA*, April 2003.

[ZBS98] Z. Zhang, V. Basili, and B. Shneiderman. An empirical study of perspective-based usability inspection. *Proceedings of the Human Factors and Ergonomics Society 42nd Annual Meeting, Santa Monica, CA, USA*, pages 1346–1350, October 1998.

[ZGPFM07] A. Zubiaga, A.P. García-Plaza, V. Fresno, and R. Martínez. Content-based clustering for tag cloud visualization. *Universidad Nacional de Educación a Distancia (UNED), Madrid, Spain*, 2007.