2013 22nd Australian Conference on Software Engineering

From Toy to Tool: Extending Tag Clouds for Software and Information Visualisation

Jessica Emerson, Neville Churcher and Chris Deaker
Department of Computer Science and Software Engineering, University of Canterbury, Private Bag 4800
Christchurch 8140, New Zealand
Email: neville.churcher@canterbury.ac.nz

***Abstract*—Software visualisation employs techniques from the more general information visualisation field to help software engineers comprehend and manage the size and complexity of software systems. The scale and complexity of the software engineering domain pose significant challenges and it is important to make effective use of techniques which can be adapted effectively to support tasks in this context. In this paper, we extend significantly the tag cloud concept, transforming it from a simple toy into a powerful tool which can help address challenges inherent in software visualisation. We illustrate our approach with examples drawn from our software engineering research programme and describe TAGGLE a tool which implements our techniques. Our visualisations support developers as they search, filter, browse, explore and act upon data and are a useful addition to the software visualisation tool kit.**

I. INTRODUCTION

Despite many advances in theory, tools and practices, scaling software development to cope effectively with systems of ever-increasing current sizes and complexities has continued to be very demanding — as is indicated by the number of IT projects which fail or are significantly compromised. The problem is multi-faceted: more than just technical aspects are involved. Developers need to communicate with their future and former selves: diagrams and visualisations provide convenient *time capsules* for this purpose.

Software visualisation, a sub-field of information visualisation, involves quantities which have no intrinsic geometric representation: computed geometry based on an underlying metaphor underpins individual visualisations. The chosen metaphor has a large influence on the success or otherwise of a visualisation and some general principles have emerged: examples include *exploit human perception strengths*, motivating techniques such as Chernoff faces [1] and geons [2], and *limit the number of data items to be managed concurrently* which leads to the "magic number" 7 ± 2 [3].

There is no single right way to go about designing a visualisation: the overall effectiveness of a metaphor, and the specific techniques used to realise it, depend on factors such as scale, specific tasks and individual variations.

Scale: a visualisation needs to be both tractable computationally and comprehensible effectively by individuals as data set sizes may range over several orders of magnitude [4].

Task: as a user explores a data set the visualisation itself may need to evolve in order to best represent the user's changing needs as quantities such as time and working set change. This is particularly evident in visualisation of dynamic systems (e.g. [5]).

Individual: User differences resulting from factors such as age, experience, or gender can be significant. These factors could affect the assumed levels of dexterity, interface complexity and colour blindness respectively. Human perception is also a factor (e.g. we are better at distinguishing size differences than colour differences) to be considered [6].

A large number of visualisation techniques have been developed and applied [7]–[10]. However, there remains the need to explore new techniques, to evaluate their strengths and weaknesses, and to use them effectively to improve the overall effectiveness of visualisations. In particular, we need to consider carefully how to adapt existing techniques to cope not only with the scale and complexity of data sets in the software engineering domain but also with the nature of the tasks performed by software engineers.

We have previously developed tools for specifying, measuring and visualising properties of software systems. Our work has included both "hard" metrics and "soft" heuristics [11]–[13].

We have used a pipeline-based approach in much of our software engineering work [14], [15]. In the visualisation pipeline, data undergoes a series of processes and transformations (initial capture, selection, filtering, geometry computation, ...) leading to the eventual delivery of a visualisation to users. In earlier work we addressed issues pertaining to the design of pipeline-based software visualisations, and illustrated the use of a parallel design pipeline in the design and evolution of visualisations [16]. We use tag clouds as one of many available techniques for presenting the visualisation to the user for analysis, exploration and action.

Tag clouds are a relatively recent addition to the information visualisation toolkit. While they are appealing and appear to have potential applications in information visualisation and software visualisation, these have not been explored fully or applied effectively.

In this paper, we extend the tag cloud concept in order to accommodate information visualisation features including those specifically relevant to software engineering data. We consider some of the issues which characterise software visualisation as well as more general information visualisation contexts. We describe TAGGLE, our prototype implementation, and present examples drawn from our software engineering research. Our work is motivated by the desire to extend tag

978-0-7695-4995-8/13 $26.00 © 2013 IEEE
DOI 10.1109/ASWEC.2013.27

Fig. 1. Tag cloud summarising this paper's content

clouds from a simple toy to a powerful tool which is well suited to the visualisation tasks performed by software engineers.

The remainder of the paper is structured as follows. In the next Section, we give a brief overview of tag clouds and their potential for visualisations. Related work is summarised in Section III. We describe our approach, together with the tool we have implemented, in Section IV and illustrate its support for visualisation tasks with a number of examples from the software engineering domain. Some issues relating to scaling are covered in Section V. Sections VI and VII summarise our results, conclusions and ongoing work. Larger versions of figures from the paper are available at http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle.

II. Extending Tag clouds for visualisation

Tag clouds have become commonplace. The concept of "subconscious files" as weighted lists of keywords occurs in Douglas Coupland's 1995 novel Microserfs [17] but Wordle (http://www.wordle.net) really popularised the concept. They are often used to indicate the content of blogs and have become popular in other information retrieval contexts where the content of a document or document collection is of interest. Figure 1 shows a typical example — Wordle's representation of the text of this paper.

Tags (text labels for data points) are arranged according to some layout strategy. The font sizes used generally correspond to the frequency with which the tags appear in the document under consideration. Conventional tag clouds typically limit themselves to showing the frequency of tags in the underlying data and limit tag numbers (Wordle's default is 150).

Tag clouds are often regarded as 'toys' because of the contexts in which they appear and the simplicity of the basic technique and its implementations. We aim to show that tag clouds can be extended to become a powerful tool alongside other information visualisation techniques and that they can be applied effectively to the software engineering domain.

In previous work we have begun to explore the potential to extend the tag cloud metaphor to enable clouds to be used as an effective visualisation tool [18], [19]. The key idea involves mapping variables in the data set to visual attributes of the tag cloud. In this way, attributes such as colour may be used to convey additional information rather than just function as "decoration." A key concept in our approach is to support flexible mappings from variables in the data set to visual attributes of corresponding clouds. Similarly, we extend the basic concept by providing facilities for interacting with clouds.

The models which are the subjects of software visualisations essentially consist of elements, properties of elements and relationships between elements. Users of the visualisations perform a range of navigation, comprehension and exploration tasks involving:

- Gisting or overall impression forming
- Determining the presence/absence of an element, property or relationship
- Studying the distributions of quantities
- Exploring correlations and relationships between quantities
- Discovering new knowledge from the data

Tag clouds can help with each of these.

In the software visualisation domain, data distributions (such as a histogram of LOC) are typically heavily skewed, may contain large numbers of data points, may cover large value ranges and may include outliers [20]). Outliers may actually be the most interesting data points (e.g. a method which is suspiciously complex for its size) and thus can not be ignored.

Techniques for handing large amounts of data may not themselves necessarily provide a good balance of focus and context information. Combining techniques can allow visual indexing and other metaphors to be applied. No single technique is universally ideal: the challenge is to find "horses for courses" using individual techniques in contexts where they are most effective.

We see tag clouds fitting alongside multivariate data alternatives such as kiviat charts, treemaps and parallel coordinates [9], [10].

Another key challenge is the focus+context problem, where drilling in to explore detailed properties of specific components leads to the user losing awareness of the relationships with the remainder of the system.

Whatever technique is used, we need to be able to map sufficient variables to be useful in software engineering contexts and to support users in the information analysis tasks they perform.

We have previously considered the design of software visualisations [16] and make use of that approach in this work.

Naïve application of tag clouds to software engineering data sets demonstrate the potential limitations of the technique and the opportunities for improvements.

There are very few existing tag cloud implementations which support tag cloud visualisation of source code or software metrics data. One is the Sonar Software Analysis Platform (http://www.sonarsource.org which includes a basic tag cloud component which we used to produce the cloud shown in Figure 2 for Aspectj — an open source project containing 2,342 classes.

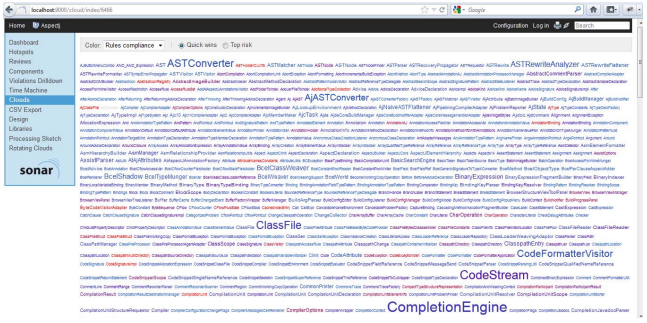


Fig. 2. AspectJ visualised in Sonar

The tag font size is mapped to LOC (Lines of Code) and the tag colour is mapped to a rules compliance metric. The (unqualified) class name is used as the tag identifier — which will prove problematic if class names are not unique.

This cloud is not "nice" — the large number of tags limits readability and there are "rivers" of white space. Even so, it is possible to identify readily the largest few classes and to infer something about the LOC distribution. Individual tags function as hyperlinks to more detailed information about the corresponding classes.

III. Related Work - Extensions and Evaluations

Some studies have been done on the evaluation of tag cloud usage but little of this relates directly to the activities specific to our software engineering research. While the general principles have informed our approach, our own empirical studies will be required.

Rivadeneira et al. conducted an early empirical tag cloud evaluation, and proposed a set of user tasks supported by tag clouds [21]. These include searching, browsing, impression forming and recognition/matching. Font size has been discovered to have a strong effect on user perception [21]–[23], although Bateman suggested this effect is influenced by other visual properties. Evaluation of eye-tracking data supports this suggestion [24].

It has been noted that tags in the upper left quadrant of a tag cloud are found more quickly [22] and are better recalled [21]. Analysis of eye-tracking data has shown the upper left quadrant of a tag cloud receives the most attention [24], [25].

Studies [22], [23, for example] have suggested users 'scan' tag clouds rather than read them and research considering visual exploration strategies of tag clouds and recorded gaze data has supported this finding [24], [25]. Schrammel et al. also found evidence for two types of user search strategy — chaotic and serial scanning. Users typically switched to serial scanning (performed in a characteristic zig-zag pattern), after a chaotic search pattern did not yield a result.

Hearst and Rosner [26] investigated the motivations behind tag cloud usage. Results from their interviews indicated that users perceive a primary advantage of tag clouds as signallers of the social activity occurring within a system.

Oosterman and Cockburn [27] found that tables were faster and more accurate than tag clouds for some specific element identification tasks. Others [21], [23], [28] found results indicated ordinary lists perform better than tag clouds for user search tasks, although reported user satisfaction was higher with tag clouds [28].

Users preferred a traditional search interface for specific information retrieval tasks but the tag cloud was preferred where the information seeking was non-specific. They concluded tag clouds were worthwhile as a supplement to traditional interfaces rather than a replacement [29].

Improvements to tag cloud construction and layout have been suggested [30], [31]. Semantically clustered tag clouds for search and recall tasks have also been evaluated [32]. Results indicated that for specific search tasks, semantically clustered tag clouds could improve performance over random layouts, although not alphabetical layouts. Users seem able to more easily find tags in alphabetically ordered clouds [23].

Modified tag clouds have been proposed with various applications. Fujimura et al. presented an algorithm for displaying large datasets in a tag cloud [20]. A layout algorithm to display a hierarchy in a way that captures contextual relationships between tags and promotes navigation has been suggested [33] as have coupling trend charts with word clouds to illustrate temporal content evolutions in a set of documents [34], investigating semantically informed navigation within a tag cloud [35], evaluating tag cloud usage for image retrieval [36] and integrating sparklines into tag clouds to convey trends between multiple tag clouds [37].

Much of this research may be drawn upon when considering the applications of tag clouds or other visualisations in the software engineering domain. For example, the identification of corresponding software engineering tasks to searching, browsing, impression-forming and recognition/matching as classified by Revanedeira et al. [21]. The analysis of eye-tracking data showing user preference for the upper left quadrant of a visualisation is also interesting, since popularly used software diagrams such as UML class diagrams, tend not to use this space extensively.

However, we need to be aware that there are several issues, some of which were noted in Section II, specific to the data sets occurring in software visualisation which must be considered when taking these studies into account.

IV. Tag clouds with Taggle

In this section we describe the design, implementation and use of Taggle. Further detail may be found elsewhere [19]. Taggle serves both as a prototype for users and as a platform for ongoing research into layout algorithms and other areas. Taggle is written in Java and makes particular use of the Java 2D API.

We illustrate many of the key ideas using examples from the software engineering domain in order to indicate Taggle features in context.

Data is maintained in XML files conforming to a DTD appropriate for input to Taggle. This approach allows transformation to and from other common formats — such as that used by ggobi (http://www.ggobi.org) and other tools — and facilitates our pipeline approach [14]–[16]. Taggle's internal

data structures provide an API allowing direct connection to application data sources. This is particularly useful where the data set is evolving during a TAGGLE session.

Users "design" clouds by selecting values for the parameters which determine layout, mappings and other properties. Settings can be saved and loaded to allow re-use of particular combinations with different data sets. TAGGLE then takes care of the extraction of the required data, application of mappings and rendering of the cloud. Clouds can be saved and translated to other formats, such as SVG, as required.

Users may select tags as targets for subsequent actions (e.g. removal, new cloud creation, ...), obtain more detail about individual tags, observe areas at higher magnification and so on. Depending on the nature of the operation, the cloud may need to be re-computed and re-rendered.

In our work to date, most of our data sets have been taken from our software metrics projects. For example, we might generate a cloud whose labels are class names, font sizes indicate WMC[1], font colours indicate CBO, font transparency indicates proportion of public elements and font style indicates whether the class is abstract or concrete.

We have implemented several layout algorithm families. Tags are placed one at a time, with the selection order (ascending or descending) and other parameters being chosen by the user. Further algorithms can be plugged in as required. Currently implemented algorithms include:

Typewriter: Tags are placed left to right, skipping to a new line when the next tag cannot be placed on the current line.

Spiral: The first tag is placed at the centre of the region, with successive tags being placed around it in a spiral pattern. If the current tag cannot be placed in the first candidate position then several attempts may be required before a suitable location is found. The unsuitable locations will then be used as candidate locations for subsequent tags.

Force-directed: Relationships between tags are used to promote clustering of tags which "belong" together. Each cluster is initially laid out using a spiral layout. A force-directed placement model is used to determine the ultimate layout [15], [39].

Examples of these algorithms in action are presented in §IV and further information is available elsewhere [19]. Videos showing the placement process in more detail may be seen at http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle.

Figure 3 shows some typical examples from a session with TAGGLE. The data set contains 75 (public, non-constructor) methods from a Java application.

Users specify *mappings* from variables in the data set to visual attributes of the rendered cloud. Figure 3(g) shows detail of the font size mapping. In this example the font size is determined by the value of statement count (a LOC variant) for each method, mapped via a linear transformation to sizes in the 10–36 point range. A non-linear mapping is available for use when the data values have very large ranges. Similar configuration options are available for font colour, transparency, font family and style.

[1]WMC and CBO are two of Chidamber and Kemerer's OO metrics [38]

Figure 3(a) shows a cloud resulting from the mappings *statements* $\mapsto$ *font size*; *cyclomatic complexity* $\mapsto$ *colour*. Colour chips at the top right indicate the legend corresponding to the current colour mapping and current mappings are summarised in the status bar at the bottom of the window (visible in Figures 3(i) and 3(j)).

A typewriter layout with tags ordered by label has been specified. Alphabetical orderings are particularly effective for tasks involving searching for a specific tag (or confirming its absence). In this ordering the variation in font size leads to areas of white space as successive lines are "forced apart" by a large tag.

Figure 3(b) shows the same data with the mappings changed to *statements* $\mapsto$ *colour*; *cyclomatic complexity* $\mapsto$ *font size*. Comparing clouds based on the same data set but with different mappings is straightforward. A feature is provided to allow duplication of a cloud and a new cloud can also be formed from selected tags.

Since we perceive visual attributes differently (e.g. we are generally better at distinguishing variations in size than in colour or transparency and individuals also exhibit variations in perception) the ability to switch mappings conveniently is important for tasks involving correlation or other relationships.

Figures 3(e) and 3(f) show the same mappings with the tags ordered by cyclomatic complexity and LOC respectively. The relative position of a tag in these two clouds allows its position on a corresponding scatter plot to be estimated.

Individual tags may be examined in more detail as required. Figure 3(c) shows the detail of all fields for the tag labelled next_token — including those not mapped to any visual attribute.

A region may be magnified as shown in Figure 3(d). We envisage extending this simple facility to a form of bifocal display [40].

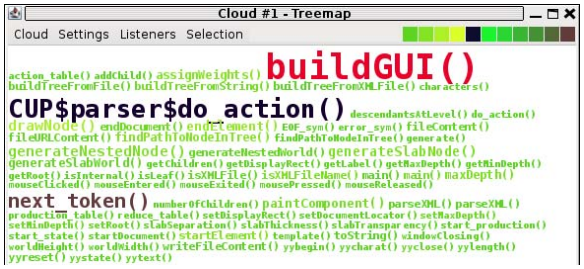
Selecting tags, individually or as a region, allows users to:

- move the selected tags to a desired position manually
- remove the selected tags from the current cloud
- retain the selected tags and remove the others
- create a new cloud containing the selected tags.

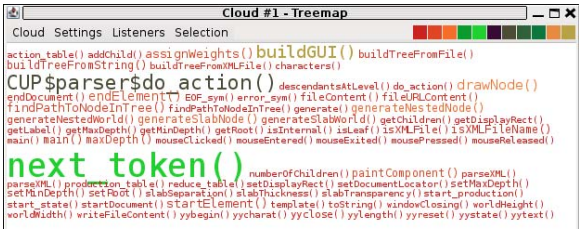
However, it is also useful to be able to filter out tags matching specified criteria. We currently have a simple expression-based filtering mechanism, shown in Figure 3(h). The expressions may involve variables which are not currently mapped, a useful way to explore higher dimensional data sets.

Figures 3(i) and 3(j) show the same data set with a spiral layout selected. These layouts are less suitable for tasks involving locating a specific tag (or confirming its absence) but allow closer packing, avoiding issues of irregular white-space between rows as experienced in typewriter layouts (such as that of Figure 3(b)).

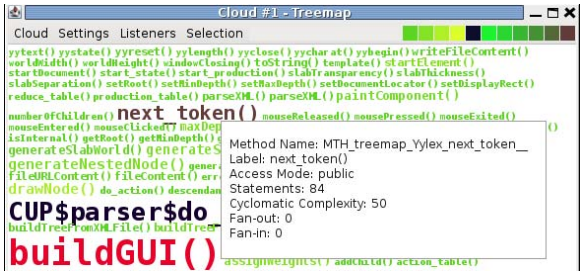
We have found tag clouds to be well suited to tasks such as search, overview and comparison. However, as for many other other techniques with similar strengths (such as SeeSoft [41]) they do not initially appear to be so well suited to tasks



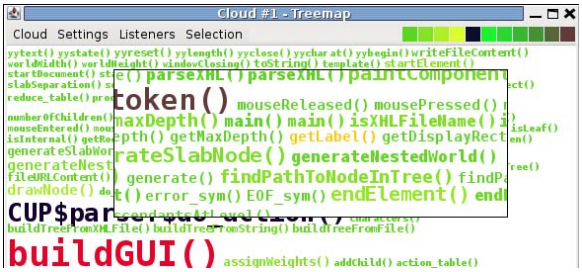
(a) Ordered by label, no filtering



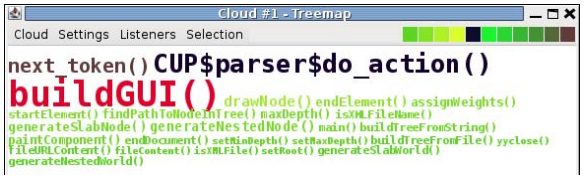
(b) Ordered by label (reverse), CC ↦ size, LOC ↦ colour



(c) Tag detail

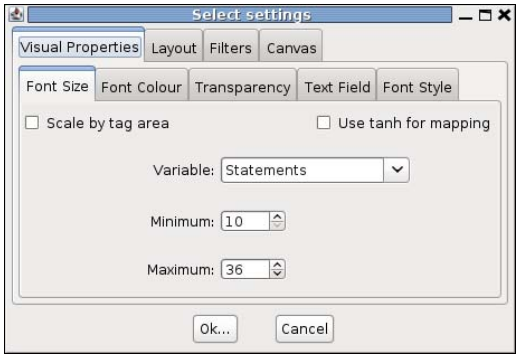


(d) Region zoom

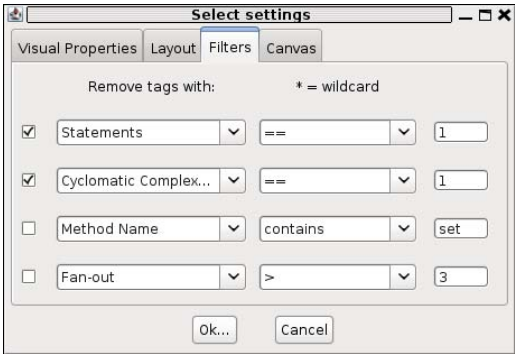


(e) Ordered by CC descending

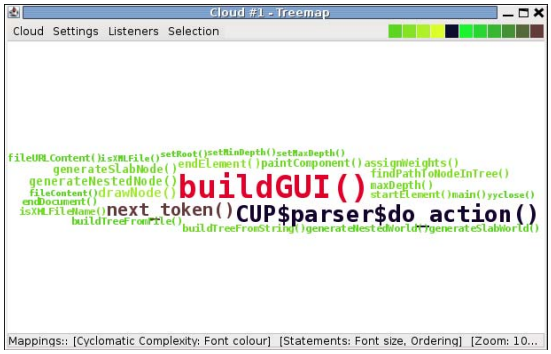
(f) Ordered by LOC descending



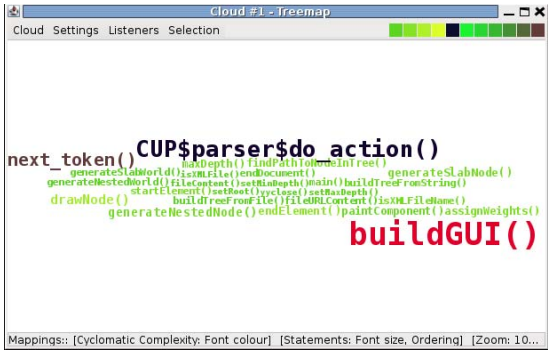
(g) Font size mapping



(h) Filter selection



(i) Spiral, ordered by LOC descending

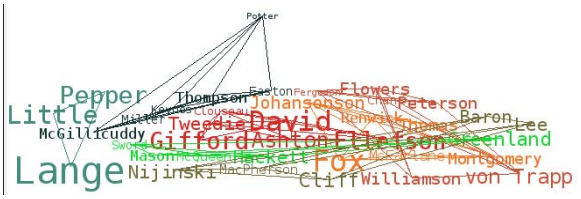


(j) Spiral ordered by LOC

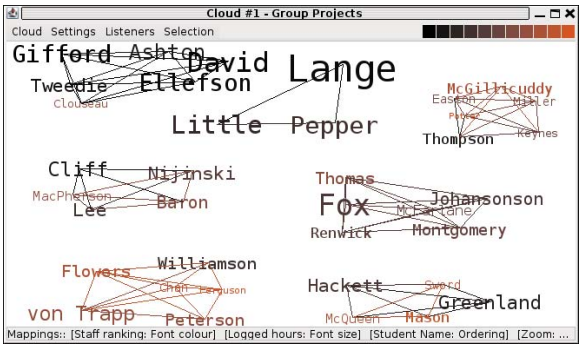
Fig. 3. Java application examples

(a) logged hours ↦ size, group name ↦ colour

(b) Partnership relationships



(c) Moving tags



(d) Emphasizing clusters with force-directed layout

(e) Clouseau's partners

(f) Linking clouds

Fig. 4. Group project examples

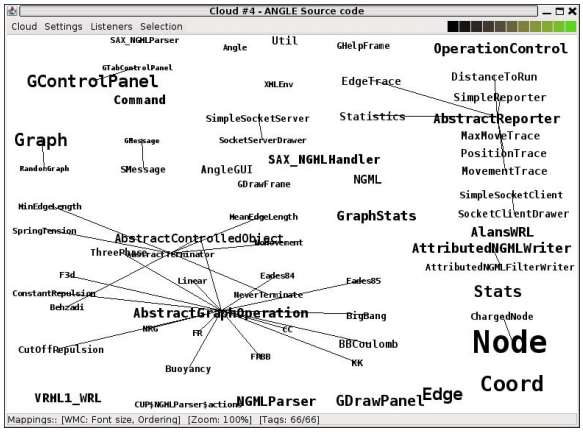


Fig. 5. Using force directed layout to explore inheritance relationship structure

involving the exploration of complex relationships between variables. We address this in a number of ways.

We present some examples based on data from a software engineering group project course we teach (names have been changed). Students log the hours they work on project activities. An agile process is used and at the end of each iteration we collect both peer assessment and marks awarded by staff. Variables in this data set include the student name, group name, mean (peer & staff) marks, and rank in class as determined by peer and staff marks.

Figure 4(a) shows tags for the 37 students in the course with mappings *logged hours* ↦ *font size* and *group name* ↦ *colour*. The cloud layout is spiral and ordered by group (i.e. successive tags to be placed are chosen by group). This leads to a layout where groups whose members' tags are placed relatively early will be clustered together, while later groups will be less clustered because some tags won't be able to be placed in the first candidate position.

Individual tags, or sets of tags, may be highlighted and actions may be bound to the highlighted tags. The default binding allows selected tags to be moved. This allows the user to "park" tags to the side for later examination or to explore the cloud using the initial layout as a starting point.

TAGGLE's data format allows the specification of relationships between tags. In this example, we have included the partnership relationships between students in a group. These relationships may be displayed on the cloud (see Figure 4(b)) but this can lead to excessive clutter. Moving one or more tags allows connections to be seen more clearly. Figure 4(c) shows the result of moving the tag labelled 'Potter', initially placed centrally (at right of 'David') to make its connections more obvious.

The overall relationship structure may be also explored in detail by selecting a force directed layout. This layout algorithm is based on the spring embedder model [39] and our own variations [15]. In this approach tags repel each other and are pulled closer to related tags, while not wanting to stray too far from the cloud centre or too close to the region boundary. Balancing these forces leads to a layout corresponding to a state in which these competing forces are balanced in an

acceptable equilibrium. The various factors contributing to the forces may be configured for finer control over the layout using GUI controls.

Figure 4(d) shows a force directed layout. The mappings are *logged hours* $\mapsto$ *font size*; *staff ranking* $\mapsto$ *colour*. Relationship lines indicate group membership and a force layout has been used in order to separate groups more clearly. This approach starts to move a little away from the classical tag cloud in the direction of general graph layout. Nevertheless, the ability to switch layout algorithms, variable mappings and filters with minimal effort adds considerable flexibility to the approach.

Figure 4(e) illustrates the use of highlighting and linking. In Figure 4(e) font size indicates the number of hours logged by each student. The cloud layout is spiral and ordered by group. Some tags have been highlighted (the current binding uses colour to indicate highlighting but custom extensions are provided for).

TAGGLE allows a *highlight listener* to be added. When a tag is selected, the listener highlights the *related* tags. This is another powerful exploration tool, and provides facilities similar to *brushing* as found in tools such as ggobi.

Highlight listeners may be added to *other* clouds. This enables groups of clouds to be used in a manner reminiscent of a scatterplot matrix or other environment where multiple information graphics are available. This is a particularly effective way to explore more than one relationship concurrently. The other clouds are typically created by duplicating the current cloud or from a selected subset of the current cloud's tags. Different mappings will generally be selected in order to explore more relationships.

Figure 4(f) shows the same tags with font size now mapped to grade awarded by staff and no colour mapping. The layout here, a typewriter layout with student names in alphabetical order, makes it easy to find a given student while removing any group clustering.

Highlight listeners have been selected so that when a tag representing a student is selected, the student's group partners are highlighted in both clouds. Figures 4(e) and 4(f) correspond to selecting the tag 'Clouseau' in Figure 4(f). Thus we see that although Figure 4(f) shows that Clouseau has a lower staff grade than his partners, Figure 4(e) suggests that this might be because he has logged fewer hours than his partners.

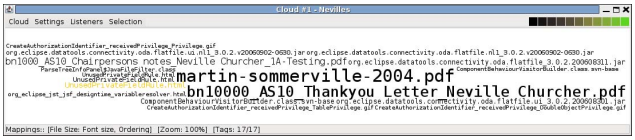
Figure 5 shows another force directed layout containing 66 tags corresponding to classes in a Java application. Font size indicates their WMC metric values and edges indicate inheritance relationships. Linking via highlight listeners to related clouds (such as those of Figure 3) allows exploration.

V. SCALING — PUSHING THE BOUNDARIES

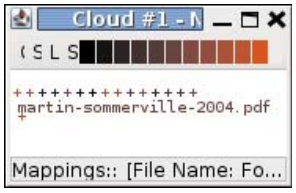
Any visualisation technique comparable to tag clouds for information visualisation purposes will eventually encounter limits as the dimension of the data set (number of variables) and/or the number of data points increases. Sometimes further limitations arise because of user characteristics (e.g. our limited ability to distinguish differences in transparency). In this section we describe how we deal with such issues in TAGGLE.

(a) Long tags

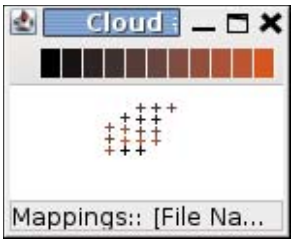
(b) Constraining tag length by truncation



(c) Fully truncated



(d) Shrinking window (or adding many tags)



(e) All tags collapsed

Fig. 6. Strategies for coping with structured and oversize tags

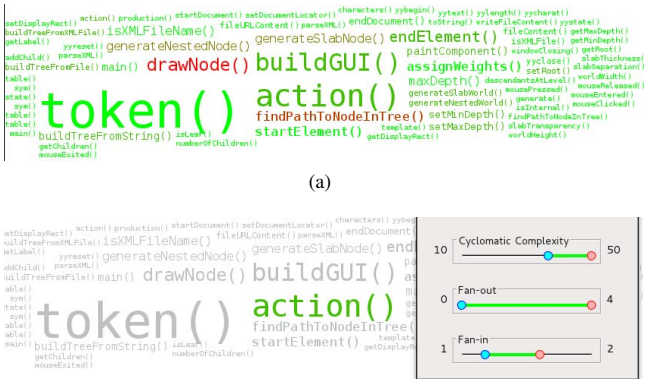
Figure 6 illustrates some of the challenges to tag clouds as the boundaries are pushed to more realistically represent data likely to occur in software visualisation applications. These challenges are by no means unique to tag clouds as a visualisation tool, but do need to be addressed if they are to be widely applicable.

One of us (N.C.) wondered how many of his files had names containing the letters of his own name in order. Figure 6 shows the file names returned by the corresponding Unix `find` command with font size mapped to file size.

As can be seen in Figure 6(a), some of the file names are very long. Long tags are problematic. They are clearly hard to place — particularly if they are longer than the width of the tag region. This situation may arise if the tag label is long (as in Figure 6(a)) or where the window size is reduced by the user.

A further difficulty is that when tag lengths vary considerably, a long tag appears to "claim" more real estate in a cloud than it deserves since even if the font size is very small it will still require more space than a shorter tag. We have explored algorithms which constrain tags' aspect ratios as one way to deal with this though we are not convinced that this is the best solution.

If a tag cannot be placed because it exceeds the width of the available space then it is replaced by a glyph — a '+' in



(a)

(b)

Fig. 7. Dynamic query filtering

Figure 6(a). The details of such collapsed tags may be examined, as is shown for `UnusedPrivateFieldRule.html` in Figure 6(a), in order to decide whether filtering, resizing the window or other actions are appropriate for the task in hand.

In our software visualisation work, long tags tend to be structured implicitly, with a delimiter such as '/' or '.' separating the levels. Examples include file/path names, URIs, fully qualified identifiers and mangled identifiers. We have implemented a feature which allows successive truncations based on the delimiter specified. Figure 6(b) shows the cloud of Figure 6(a) after 4 truncations with the delimiter '/'. Note that the tag `UnusedPrivateFieldRule.html`, with the leading `/home/cosc/staff/` having been removed, is now able to be placed (shown highlighted in Figure 6(b)). Figure 6(c) show the fully-truncated version in which all tags are able to be placed.

Figure 6(d) shows the effect on the layout of Figure 6(c) of reducing the window size. The algorithm has reduced the font size in order to place as many tags as possible (only 1 at this point) and the others are all collapsed. The effect of reducing the size even further is shown in Figure 6(c). Even such low level of detail can still be helpful in visualisation tasks: properties such as tag colour remain visible and tags may still be selected and moved.

Figure 7 shows another powerful mechanism for exploring large clouds. We have implemented a form of dynamic query [42]. This allows active filtering based on values of all variables — whether or not they are mapped to a display attribute such as colour or size. The filtering does not change the layout (though there are other facilities to create a new cloud from selected tags) but the filtered tags may be dimmed or removed altogether in order to highlight others. For numeric variables the minimum and maximum values are specified via slide controls and text expressions are used for other fields.

Figure 7(a) shows a cloud of 75 method names from a Java package with mappings *cyclomatic complexity* $\mapsto$ *font size*, *fan-out* $\mapsto$ *colour* in a spiral layout where methods with placement order determined by descending LOC value. Figure 7(b) illustrates the result of limiting the range of the cyclomatic complexity (mapped to font size) and fan-out (unmapped) — only one tag remains and is clearly distinguished from buildGUI() which otherwise has very similar visual attributes.

We have found this approach to be sufficiently flexible and powerful to deal with a range of data sets. Ultimately, when limits are encountered we can still gather sufficient information to allow switching to alternative techniques where necessary. This is a common task pattern in information visualisation: one technique (tag clouds) is used to filter and explore higher "big-picture" levels of context and others (such as tables) are used to obtain detail of selected data items.

VI. Discussion & Empirical Issues

Having used TAGGLE on a wide range of data sets, we have formed some general views on the relative merits of tag clouds and our TAGGLE prototype application.

How many tags can be managed effectively? We believe that tag clouds are best suited to tasks where the number of tags to be displayed at any one time is no more than about 50. In practice "right sizing" a cloud involves factors such as:

- filtering to remove tags of currently low relevance.
- tag label choice. A 'short name' variable, or coding, can often be chosen if there is no obvious delimiter to use for truncation.
- allowing sufficient real estate (dimensions and aspect ratio) to allow cloud layout algorithms sufficient headroom.

TAGGLE provides us with a platform for experimenting with layout algorithms. It has not been highly optimised, but performance is satisfactory for our purposes — the largest clouds shown in this paper took no more than a few seconds to lay out. However, our current spiral layout algorithms could benefit from re-factoring in order to improve performance for large clouds. Since very large clouds are not particularly comprehensible without filtering or other reduction techniques, such limitations have not proved problematic thus far. The extensible nature of TAGGLE supports the development of new algorithms, visual attribute mappings and interaction mechanisms.

Some studies, such as those mentioned in Section III, of the usability and efficacy of tag clouds have been made. In some cases, only typewriter algorithms are considered. In others, the data sets are artificial or familiar to the users. The variation evident to date means there is no clear standard to compare with. Nevertheless, there are some aspects (e.g. larger tags attract more user attention) where consensus has been achieved and these provide us with guiding principles for our continued work.

While these issues affect the absolute value of tag clouds, we must also consider their merits relative to alternative techniques for a given task. We are interested in issues such as whether users "read" a cloud by scanning left-to-right, top-to-bottom and the effect of the distribution of tag label lengths. These are issues of wider significance. There are also issues specific to software visualisation which we wish to explore.

To supplement our own experiences and anecdotal feedback from our colleagues and students we are currently conducting experiments to shed more light on these issues. A Tobii eye tracker (http://www.tobii.com), captures for later analysis data about where the user is looking while performing a task.

Figure 8 shows some indicative results obtained by tracking a user (NC) over a period of approximately 10 seconds as he looked at the cloud shown in Figure 8(a). The tags represent 25 Java classes, with size and colour mapped to metrics from the Chidamber & Kemerer suite.

Figures 8(b)–8(d) show *heatmaps* at $0.5s$, $4.9s$ and $10s$ respectively. Regions where the user's attention spends the most time appear in red; green indicates regions given relatively little attention.

Figures 8(e) and 8(f) show *gaze maps* at $0.5s$ and $10.1s$ respectively. The circles represent periods where the user's gaze is focused for a time longer than the current threshold: they are numbered in sequence and the lines show the direction of eye movement between successive attention foci.

Video clips showing the development of these patterns in greater detail are available at http://www.cosc.canterbury.ac.nz/research/RG/svg/taggle together with information about the experimental configuration.

VII. Conclusion & Future Work

In this paper we have demonstrated that tag clouds can be extended to include additional information visualisation functionality. This enables tag clouds to be used effectively in software visualisation.

We have described a tool, TAGGLE, which implements our extended tag cloud models. We are encouraged by our experiences thus far, and are continuing to develop our techniques further. One current project involves include tag clouds in standard javadoc HTML pages.

We have used TAGGLE on data from software engineering projects; typical variables are class names, method names, access modes, and software metrics such as lines of code, cyclomatic complexity and the Chidamber & Kemerer suite. Using the mapping mechanism, with its linear and non-linear (tanh) transformations has allowed us to gain insight into our data sets as effectively as other techniques we have used.

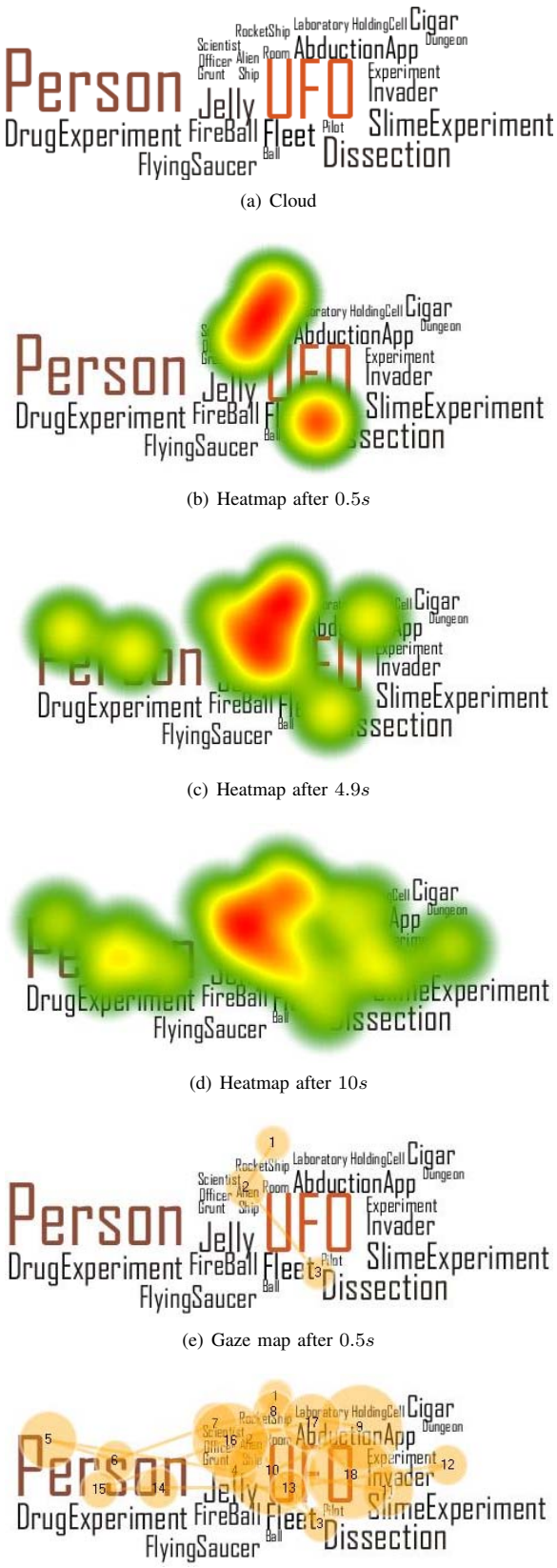
As well as being usable in its current form, TAGGLE provides an extensible platform for ongoing exploration of layout algorithms, mapping techniques and other aspects of our software visualisation research programme.

Evaluation is a vital, but challenging, part of visualisation. Our department has recently obtained a Tobii eye tracker (www.tobii.com) and we are currently engaged in empirical studies to help us both quantify the effectiveness of the current implementation but also to help us determine the most appropriate software visualisation tasks to deploy tag clouds on. Such empirical work complements our ongoing feedback from users and we envisage further applications of tag clouds in our own work.

References

[1] H. Chernoff, "The use of faces to represent points in k-dimensional space graphically," *Journal of the American Statistical Association*, vol. 68, no. 342, pp. 361–368, 1973.

[2] P. Irani and C. Ware, "Diagramming information structures using 3d perceptual primitives," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 10, no. 1, pp. 1–19, 2003.



(a) Cloud

(b) Heatmap after $0.5s$

(c) Heatmap after $4.9s$

(d) Heatmap after $10s$

(e) Gaze map after $0.5s$

(f) Gaze map after $10.1s$

Fig. 8. Tracking the Aliens

[3] G. A. Miller, "The magical number 7 plus or minus two: Some limits on our capacity for processing information," *Psychological Review*, vol. 63, pp. 81–97, 1957.

[4] A. Unwin, M. Theus, and H. Hofmann, *Graphics of Large Datasets: Visualizing a Million*. Springer, 2006.

[5] J. Ashford, N. Churcher, and W. Irwin, "Dynamic visualisation of software state," in *Australasian Computer Science Conference (ACSC 2011)*, Perth, Australia, 2011, pp. 127–136. [Online]. Available: http://crpit.com/confpapers/CRPITV113Ashford.pdf

[6] W. Cleveland, *The Elements of Graphing Data*. Hobart Press, 1994.

[7] S. Card, J. Mackinlay, and B. Shneiderman, Eds., *Readings in Information Visualisation: Using Vision to Think*. Morgan Kaufman, 1999.

[8] R. Harris, *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press, 1999.

[9] R. Spence, *Information Visualisation*. Addison-Wesley, 2001.

[10] C. Ware, *Information Visualization: Perception for Design*, 2nd ed. Morgan Kaufman, 2004.

[11] W. Irwin and N. Churcher, "Object oriented metrics: Precision tools and configurable visualisations," in *METRICS2003: 9th IEEE Symposium on Software Metrics*, 2003, pp. 112–123.

[12] B. Neate, W. Irwin, and N. Churcher, "Coderank: A new family of software metrics," in *ASWEC2006: Australian Software Engineering Conference*, 2006, pp. 369–378.

[13] N. Churcher, S. Frater, C. P. Huynh, and W. Irwin, "Supporting OO design heuristics," in *ASWEC2007: Australian Software Engineering Conference*, 2007, pp. 101–110.

[14] W. Irwin and N. Churcher, "XML in the visualisation pipeline," in *Visualisation 2001*, ser. Conferences in Research and Practice in Information Technology, vol. 11. Sydney, Australia: ACS, 2002, pp. 59–68.

[15] N. Churcher, W. Irwin, and C. Cook, "Inhomogeneous force-directed layout algorithms in the visualisation pipeline: From layouts to visualisations," in *InVis.au 2004*, ser. Conferences in Research and Practice in Information Technology, vol. 35, 2004, pp. 43–51.

[16] N. Churcher and W. Irwin, "Informing the design of pipeline-based software visualisations," in *APVIS2005*, ser. Conferences in Research and Practice in Information Technology, vol. 45. ACS, 2005, pp. 59–68.

[17] D. Coupland, *Microserfs*. HarperCollins, 1995.

[18] C. Deaker, L. Pettigrew, N. Churcher, and W. Irwin, "Software visualisation with tag clouds," in *ASWEC 2010 Industry Track Proceedings*, Auckland, New Zealand, Apr. 2010, pp. 129–133.

[19] C. Deaker, N. Churcher, and W. Irwin, "Tag clouds in software visualisation," Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand, Technical Report TR-COSC 01/11, 2011.

[20] K. Fujimura, S. Fujimura, T. Matsubayashi, T. Yamada, and H. Okuda, "Topigraphy: visualization for large-scale tag clouds," in *Proceedings WWW '08*, 2008, pp. 1087–1088. [Online]. Available: http://doi.acm.org/10.1145/1367497.1367669

[21] A. W. Rivadeneira, D. M. Gruen, M. J. Muller, and D. R. Millen, "Getting our head in the clouds: toward evaluation studies of tagclouds," in *CHI '07*, 2007, pp. 995–998.

[22] S. Bateman, C. Gutwin, and M. Nacenta, "Seeing things in the clouds: the effect of visual features on tag cloud selections," in *HT '08*, 2008, pp. 193–202.

[23] M. J. Halvey and M. T. Keane, "An assessment of tag presentation techniques," in *Proc WWW '07*, 2007, pp. 1313–1314.

[24] S. Lohmann, J. Ziegler, and L. Tetzlaff, "Comparison of tag cloud layouts: Task-related performance and visual exploration," in *INTERACT 2009*, ser. LNCS, vol. 5726. Springer, Aug. 2009, pp. 392–404.

[25] J. Schrammel, M. Leitner, and M. Tscheligi, "Semantically structured tag clouds: an empirical evaluation of clustered presentation approaches," in *CHI '09*, 2009, pp. 2037–2040. [Online]. Available: http://doi.acm.org/10.1145/1518701.1519010

[26] M. A. Hearst and D. Rosner, "Tag clouds: Data analysis tool or social signaller?" in *Proceedings of HICSS '08*, 2008, pp. 160–170. [Online]. Available: http://dx.doi.org/10.1109/HICSS.2008.422

[27] J. Oosterman and A. Cockburn, "An empirical comparison of tag clouds and tables," in *OZCHI '10*, 2010, pp. 288–295. [Online]. Available: http://doi.acm.org/10.1145/1952222.1952284

[28] B. Y.-L. Kuo, T. Hentrich, B. M. . Good, and M. D. Wilkinson, "Tag clouds for summarizing web search results," in *WWW '07*, 2007, pp. 1203–1204. [Online]. Available: http://doi.acm.org/10.1145/1242572.1242766

[29] J. Sinclair and M. Cardew-Hall, "The folksonomy tag cloud: when is it useful?" *J. Inf. Sci.*, vol. 34, no. 1, pp. 15–29, Feb. 2008. [Online]. Available: http://dx.doi.org/10.1177/0165551506078083

[30] O. Kaser and D. Lemire, "Tag-cloud drawing: Algorithms for cloud visualization," *CoRR*, vol. abs/cs/0703109, 2007.

[31] C. Seifert, B. Kump, W. Kienreich, G. Granitzer, and M. Granitzer, "On the beauty and usability of tag clouds," in *IV 2008*, 2008, pp. 17–25.

[32] J. Schrammel, M. Leitner, and M. Tscheligi, "Semantically structured tag clouds: an empirical evaluation of clustered presentation approaches," in *CHI '09*, 2009, pp. 2037–2040.

[33] K. S. Candan, L. Di Caro, and M. L. Sapino, "Creating tag hierarchies for effective navigation in social media," in *Proc SSM '08*, 2008, pp. 75–82. [Online]. Available: http://doi.acm.org/10.1145/1458583.1458597

[34] W. Cui, Y. Wu, S. Liu, F. Wei, M. Zhou, and H. Qu, "Context preserving dynamic word cloud visualization," in *Pacific Visualization Symposium (PacificVis)*, march 2010, pp. 121–128.

[35] L. Di Caro, K. S. Candan, and M. L. Sapino, "Using tagflake for condensing navigable tag hierarchies from tag clouds," in *Proc 14th ACM SIGKDD*, ser. KDD '08, 2008, pp. 1069–1072. [Online]. Available: http://doi.acm.org/10.1145/1401890.1402021

[36] J. Callegari and P. Morreale, "Assessment of the utility of tag clouds for faster image retrieval," in *Proc MIR '10*, 2010, pp. 437–440. [Online]. Available: http://doi.acm.org/10.1145/1743384.1743461

[37] B. Lee, N. Riche, A. Karlson, and S. Carpendale, "Sparkclouds: Visualizing trends in tag clouds," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, no. 6, pp. 1182 –1189, nov.-dec. 2010.

[38] S. Chidamber and C. Kemerer, "A metrics suite for object oriented design," *Software Engineering, IEEE Transactions on*, vol. 20, no. 6, pp. 476 –493, jun 1994.

[39] P. Eades, "A heuristic for graph drawing," in *Proc. 13th Manitoba Conf. Numerical Mathematics and Computing*, D. S. Meek and G. H. J. v. Rees, Eds. Utilitas Mathematica Publishing, 29 Sep.–1 Oct. 1983.

[40] Y. K. Leung and M. D. Apperley, "A review and taxonomy of distortion-oriented presentation techniques," *ACM Transactions on Computer-Human Interaction*, vol. 1, no. 2, pp. 126–160, 1994.

[41] T. Ball and S. Eick, "Software visualization in the large," *IEEE Computer*, vol. 29, no. 4, pp. 33–43, Apr. 1996.

[42] B. Shneiderman, "Dynamic queries for visual information seeking," *IEEE Softw.*, vol. 11, no. 6, pp. 70–77, Nov. 1994. [Online]. Available: http://dx.doi.org/10.1109/52.329404