Overherd Functional Specification

# Overview

Overherd is an application that allows users to visualize, explore, and read discussion boards through treemaps and social network diagrams.

The priority for features goes like this

1. Treemap view,
2. Social network view,
3. Expand data file type support.

Overherd currently supports data in its own native XML structure. That’s good enough for now. Once we get features 1 & 2 working, we can run user tests while the dev team works on feature 3.

This spec is not, by any stretch of the imagination, complete. All of the wording will need to be revised several times before it is finalized. The graphics and layout of the screens is shown here merely to illustrate the underlying functionality. The actual look and feel will be developed over time with the input of graphics designers and iterative user feedback.

This spec does not discuss the algorithms, libraries, or languages Overherd uses – they will be discussed elsewhere. It simply discusses what the user sees when they interact with Overherd.

# Scenarios

## Scenario 1: Tom the TA

Tom is a TA for a large (~200 students) Intro to Psychology class. The professor in charge of the course has decided to require all students to participate in an online discussion about their course so they can get some of that “small class” discussion feeling. She put the TAs, like Tom, in charge of managing and grading the online discussion. With 200 students posting multiple messages in multiple topics every week, it’s really hard for Tom to keep up. He doesn’t know what students are talking about. He doesn’t know who’s doing all the talking. He doesn’t know whether students are using the right concepts and understandings to have their conversation. Tom downloads his class’s discussion about once a week and imports it into Overherd. He uses the treemaps to figure out things like what conversations are recent and whether there’s a conversation getting a lot of activity. That helps him know where to look first. He uses the social network diagrams to figure out if anyone is being left out of the conversation and whether someone is doing all the talking.

## Scenario 2: Fae the Foodie

Fae loves food. She even considers herself a “food tourist”, willing to travel almost anywhere to eat great food and looking for great food any time she travels. She’s just moved to Chicago, and she wants to know where to eat all the time. She uses Chowhoud a lot because other members there are just as snobby about food as she is. Trouble is, she has to sift through a lot of crap to get to good recommendations for places to get cupcakes or who has the best Pho in town. There’s no way she would trust the yahoos over at Yelp. They think hot dogs are haute cuisine. So, Fae downloads the Chicago Chowhound board and imports it into Overherd. She uses the treemap view to see what topics are hot right now – they may be about a new restaurant like Next or the results of the Michelin ratings. She uses the social network diagram to look for her favorite members. There’s a guy name Ray who she agrees with almost all the time, so she likes to know who he’s talking to and what they’re talking about.

# Non-Goals

This version with *not* support the following features:

* GraphML input
* web-based application

# Overherd Flowchart

Here’s the general flow of action through Overherd:

# Screen by Screen

Overherd has 3 main screens

1. select data
2. treemap
3. social network

In this document, these screens will be referred to like so: Select Data. See the underline? That’s how you know it’s a specific screen and not just an action or description.

## Select Data

When the user opens Overherd, she should see a screen asking her to select a data file to display. This will look just like any other “select file” box where users can enter the full path or click “Browse” and go through those Finder-like screens to find the file. Then they can click “Go” or something to make the application display their data.

## Treemap

The Treemap is the default view and has three main sections:

1. the treemap itself
2. the detail pane
3. the user controls pane

**Technical Note**

The Squarified Treemap Layout should be used to draw the map. See Bruls M, Huizing K, van Wijk J. Squarified Treemaps. *Data Visualization 2000, Proceedings of the joint Eurographics and IEEE TCGV Symposium on Visualization*. 2000. pp. 33-42.

The Treemap screen generally looks like this:



Color in the treemap indicates depth of a message – the brighter the color, the shallower that message (i.e. each shade darker indicates another depth of reply)

Size of the boxes indicates amount of content – the bigger the box, the more content it contains. We’re going to start with “number of messages underneath” to determine size, so big boxes have lots of replies, small boxes have just a few.

**Technical Note**

Not sure whether size should be determined by word count, character count, number of replies, or what.

Not sure how to visually indicate the recency of a message.

### Treemap Pane

Each blue box represents an individual topic. Clicking on a blue box

1. displays the details of the selected message in the topic in the Detail Pane (labeled content|viewer above), and
2. Treemap Pane adjusts to highlight structurally related (i.e., through replies/parents) messages
   1. NOTE: how that highlighting is accomplished is undetermined
      1. Drop-shadow around topic bounding box
      2. 3D-ish raising of related messages
   2. selected message rendered a different color than blue (default: pink?)
3. NOTE: what happens next is undetermined
   1. REPLACED: redraws the treemap so that just those messages within that topic are shown, and all depths from that message to how ever many levels of replies it has are shown in a single screen or
   2. REPLACED: redraws the treemap so that the topic becomes the root of the treemap, and the next level of replies become the leaves. In this case, each time you click something in the treemap, it redraws so that that topic or message that you clicked is the root, and that topic’s or message’s replies are the leaves.

*If 3a, then:*

Basically, clicking in the treemap redraws a treemap of a topic instead of the whole board and lets you read the message in the Detail Pane. The second level treemap could look something like



Where messages in a topic are nested inside one another. All articles on a particular level are in a treemap, and replies are nested inside the message they reply to. In the example above, the topic is “[vim] migrating from emacs” which has 5 replies. 3 of those replies have no replies, and 2 of them do. The one at the top in dark green has 3 levels of replies.

*If 3b, then:*

The treemap will look similar to the default blue one, but the root and leaves will change.

**Technical Note**

Technically makes sense to start with 2b because it’s just iterating on an existing layout algorithm. 2a is a future feature.

### Detail Pane

The detail pane displays just that – details. Usually this means the actual content of a message and its metadata (author, time, etc.).

**Technical Note**

It should also allow users to navigate to messages related either semantically (through words), structurally (through replies), or socially (through authors). Default is structurally, and other relationships are reserved for later versions of the software.

### User Controls Pane

This pane has some controls users can adjust to change what gets displayed. The main control is a time slider that lets users control the start and end dates for messages to display – the treemap redraws showing only topics with messages inside that time constraint. The “select depth” bit in the screenshot above will be deleted.

### Other User Actions

When the user clicks on Student-Topic Viz, the Social Networkscreen opens.

When the user enters a term or phrase in the Search box and hits Enter, the treemap highlights (in pink maybe?) the threads or messages that meet the search criteria.

### Mapping Visual Properties to Message Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Message Property | Visual Property | Pane | Spec |
| Length | Size | Treemap | Directly proportional |
| Recency | Date (User Controls) | User Controls | Visible is date falls within user-specified range |
| Recency | Position | Treemap | Top left = newest  Bottom right = oldest |
| Reply depth | Color | Treemap | Darker = deeper |
| Structurally-related | Position | Treemap | Within sub-box of treemap  Highlighted when structurally related message in detail view |
| Author | Text | Detail | Author: <name> |
| Time/date | Text | Detail | Date: <time/date> |
| Title | Text | Detail | <Title> |
| Content | Text | Detail | <content> |

## Social Network

This screen shows a social network diagram where nodes are authors and edges mean the two authors have posted in the same topic. It has two panes

1. conversation
2. social network diagram

It looks something like this



### Conversation Pane

This pane shows info about the conversation – it’s keywords, the actual text of the messages in the conversation.

### Social Network Pane

Shows the social network diagram.

**Technical Note**

We can look into what algorithms to use to layout the network. I can think of times when ego networks like the one above make sense and when the network as a whole is more interesting. There are plenty of SNA libraries to help with this.