Improvements to the usability of the Web-Based implementation of PhotoCube

Jonas Sylvain Jersild Balin
Katarzyna Honorata Toborek

IT UNIVERSITY OF COPENHAGEN

IT University of Copenhagen
Denmark
14 December 2021

Abstract

This research project paper describes the M^3 data model and the PhotoCube client application for browsing large photo collections, built upon that model. Usability issues arising from using the system were identified and outlined, followed by modifying the client with new design and bug-fixes. Final usability tests indicate that the implemented changes have improved the usability, however due to methodological and statistical concerns, this cannot be safely concluded.

Contents

1 Introduction

In this paper, the web-based implementation of the PhotoCube client will be introduced, examined, and modified, with the purpose of improving the client's usability. First, the background for the Multi-dimensional Media Model data model and the web-based implementation of PhotoCube will be explored. Following this, the usability of the PhotoCube client will be examined in an exploratory usability test. Building on that, the client will be modified such that a number of the issues are rectified. Finally, there will be a usability test on the implementation post-modification, which leads to the conclusion of the paper.

The purpose of this project is twofold. First, it is to improve the usability of the web-based implementation of the PhotoCube client. Second, it is to familiarize the researchers with the data model, M3, such that they are prepared for future research and development for the virtual-reality implementation of the M^3 model (ViRMA).

2 Background

As the sizes of multimedia collections grow, managing, searching, and browsing these becomes more complex, and existing solutions are not scalable to support these needs. "Scalable Multimedia Analytics" (SMA) is a relatively new research area the purpose of which is to examine this issue.

As part of their research within this area Grímur Tómasson, in cooperation with Björn Jónsson and Laurent Amsaleg, developed the data model "ObjectCube" in 2011.[Tóm11] This model facilitates the browsing of large quantities of data in three dimensions. Another name, which is the one that will be used in this paper, is the Multi-dimensional Media Model, henceforth M^3 (pronounced "M-cubed").

PhotoCube is a prototype of the M^3 that facilitates the browsing and analysis of images hosted on an external server. The first iteration of PhotoCube, from 2011, was a desktop application coded in Python.[Sig11] In the years following, three other theses expanded upon the data model or the PhotoCube client, including adding searching functionality.

As part of his thesis, Peter Clausen developed a web browser-based implementation of PhotoCube in 2019.[Cla19] This client increased maintainability, portability, availability, and added features over the initial implementation. However, the advances came at the cost of usability, which was deemed to be lower than its predecessor's. In 2020, students at ITU further improved the web-based platform as part of their master's thesis.

Currently, the PhotoCube client is generally functional albeit not very user-friendly. The following section will detail the modern PhotoCube client and how it's built upon M^3.

3 PhotoCube and the data model

The web-based implementation of PhotoCube incorporates the main concepts of the M3 model: objects, tags, tagsets, hierarchies, dimensions and filters. First, these concepts will be consecutively described; then PhotoCube client user interface (UI) will be outlined using the M3 model concepts in a way that illustrates how and where the elements of the model are represented.

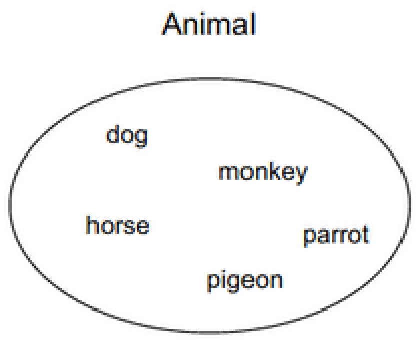
3.1 M^3 data model

In the original M^3 model an object is a piece of data, ranging through different kinds of multimedia, such as photo, video, document. However, the PhotoCube implementation works solely with photos. Each photo-object can have zero or more *tags* associated with it. For example, if a photo has a dog in it, it would be marked with the "dog" tag. These tags further belong to *tagsets*, which are sets containing various related tags grouped together. A tagset called "animal" could contain tags "cat", "dog", etc., a tagset "location" could have tags "Copenhagen", "Helsinki", etc. Furthermore, tagsets help in distinguishing between ambiguous words. "Mouse" could be either an animal or a device. Having the tag "mouse" in the "animal" or "device" tagset makes it clear what the tag refers to.

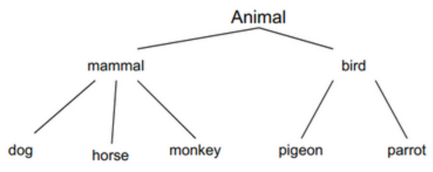
Hierarchies give structure to the tagsets. A hierarchy can have a different name from the tagset it is contained within, and it can only consist of tags from a single tagset at a time. A tagset can contain multiple hierarchies, but not other tagsets. For example, the tagset "animal" could have a hierarchy "mammal", where "mammal" at the top of the hierarchy refers to a tag "mammal" that has children "horse", "monkey", "cat". Another hierarchy within the same tagset could be "birds", with children "pigeon", "parrot", "kiwi", etc. Figure 1 presents the tagset "animal" and hierarchy "animal", which has several children which could also be separate hierarchies within this tagset.

Tagsets and hierarchies form dimensions, which organize data along axes. For example, applying a hierarchy "animal" on one axis could result in all the tags from that hierarchy being arranged alongside the axis. The photos containing these tags would be grouped in collections, displayed by their corresponding tags. Continuing the example, all photos tagged "dog" would be in a collection by the "dog" tag on the axis, similarly photos tagged "cat" would be in a different collection by the "cat" tag. If a photo contains both of these tags, it would be displayed in both groups. Projecting a certain hierarchy on a dimension means that only photos which do contain at least one of the tags from that hierarchy are shown.

There can be 0, 1, 2 or 3 dimensions, projected on the X, Y or Z axis. With



(a) Tagset "Animals"

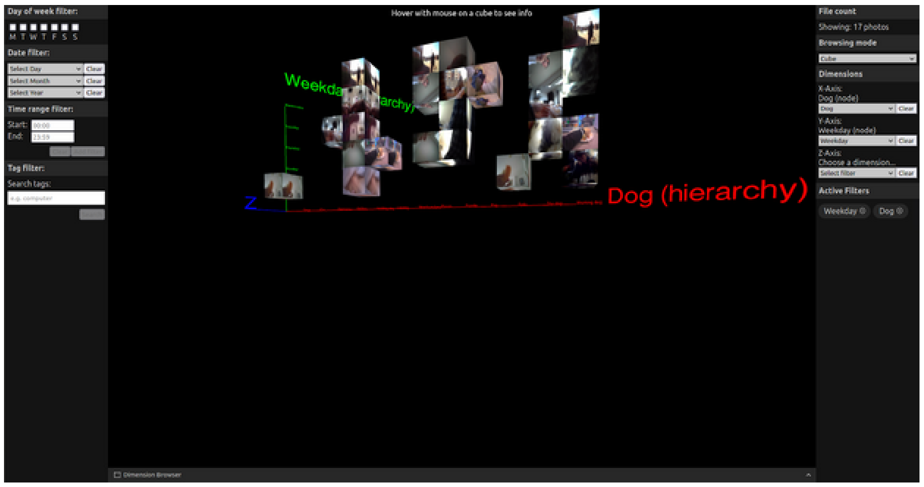


(b) Hierarchy "Animal"

Figure 1: Tagset vs Hierarchy

0 dimensions, the photos are shown in one unorganized collection. Each of the three dimensions can be viewed either vertically, horizontally or in depth. Applying filters on the axes forms a structure that can be referred to as the browsing state, which can be 1D, 2D or 3D, depending on the number of dimensions used. Figure 2 shows the PhotoCube client with 2D browsing state after projecting two filters: “dog” on the x axis and “weekday” on the y axis.

The last concept of the M^3 model are *filters*. It is important to outline the difference between the two different kinds of filters that can be used in the application. In the context of dimensions it is hierarchies and tagsets that are applied as filters. These can be described as *projected* filters, as they are being projected on the axes, as described in the previous paragraph. The second types are *direct* filters, which allow for more extensive restricting of the displayed collections. For instance, with “location” hierarchy filter applied as a dimension, applying additional tag filters “dog” and “Saturday” would narrow down the results in the browsing state to only displaying these photos with the tags within the “location” hierarchy, that also contain a dog in it and were taken on a Saturday.



Start: 00:00
End: 23:59
Showing: 17 photos
Browsing mode: Cube
X-Axis:
Dog (node)
Dog
Y-Axis:
Weekday (node)
Weekday
Z-Axis:
Choose a dimension...
Select filter
Active Filters: Weekday, Dog

Figure 2: PhotoCube browser with 2D browsing state visualisation

3.2 PhotoCube client

The PhotoCube client code and user interface (UI) are divided into three main parts: left dock, right dock and the middle. Figure 3 presents the basic view of the UI, with outlines in different colours added to the picture to highlight each part.

The middle (**Browser**) is the main part where the PhotoCube object collections are being displayed. Furthermore, it contains the bottom dock with the Dimension Browser (**DM**). The Dimension Browser in an expanded state allows searching for hierarchy and tagset filters.

Right dock (**RD**) consists of three parts: "Browsing mode", where users can choose between "Cube", "Card" and "Grid" view of the collections of photos; "Dimensions", which contains the tools for applying dimensions to the axes; and "Active filters", which displays all the filters chosen by the user.

"Cube" Browsing Mode is the default setting which visualizes the photo collections in 3D mode - referred to as the browsing state as described in the previous section. "Card" view is a simple browser that displays one photo at the time, allowing for navigation between single photos. "Grid" view populates the screen with thumbnails of photos from the browsed collection. In the browsing state dimensions can be projected on the axes by choosing a hierarchy or a tagset filter from those available under "Active filters".

The left dock (**LD**) in the UI contains several additional filters: day of week, date, time range and tag filter. When using the software, these can be applied to limit the browsing state to only contain the images, where the specified tags or parameters are present. This allows the user to refine the browsing state even further.

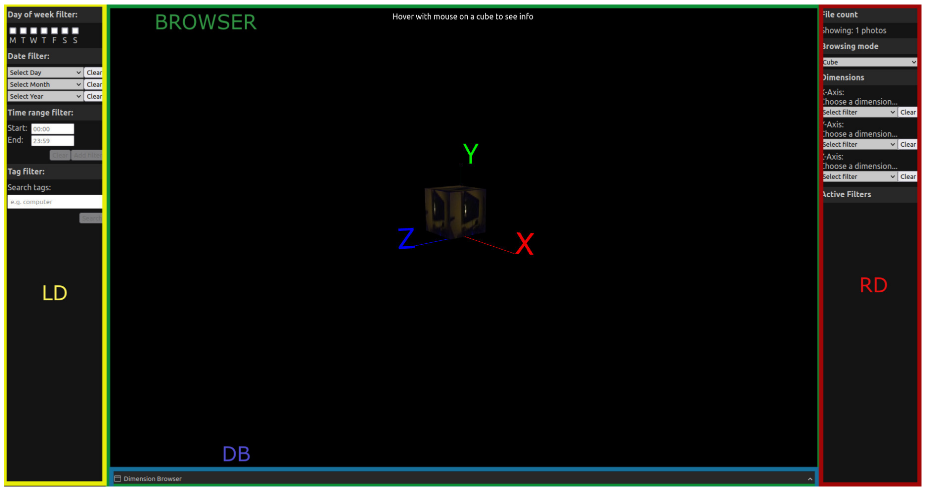


Figure 3: PhotoCube client UI

3.3 PhotoCube client code

The client code has been written mainly with React Typescript, and it currently contains elements of modern React (functional components and hooks) and "old" React (class components). The 3D mode of browsing photos - CubeBrowser, has been written using Three.js library.

The PhotoCube UI has its corresponding structure in the client code. Left dock is one of the three main components. It contains smaller parts that correspond to the elements of the left dock in PhotoCube UI - a subcomponent for each of the additional filters (day of week, date, time range, tag).

Similarly, the right dock has subcomponents for viewing active filters, changing the browsing mode between card, cube and grid, and choosing dimensions to be applied on one of the three axes.

The middle part of the UI has a slightly different structure. It contains the Bottom Dock component, which has the Dimension Browser with Tagset and Hierarchy Filter, as well as three different browsers, corresponding to Card, Grid and Cube browsing mode. Cube is the default browsing state, from which it is possible to switch between the other ones. This component is the most complex one - the aforementioned client browsing state - a visual representation of intersecting filters on the three spatial axes.

4 Preliminary explorative analysis of PhotoCube's usability

4.1 Methodology considerations

Determining usability of a software is often done by exposing it to subjects who resemble the intended user demographic. However, due to resource- and time constraints, extensive usability tests with external participants were not deemed feasible, as the project schedule also had to allow for time and study to implement solutions and a final evaluation. Beyond these limitations, it would also be highly unlikely to find participants that are representative of the hypothetical users. The PhotoCube client (in its current form) is neither designed nor ready for a wide set of users. PhotoCube's relevance is currently primarily to a specific set of academics interested in Scalable Multimedia Analytics. The issue presented by the low number of user-representative participants could potentially be mitigated by selecting non-representative participants, and then priming them on the software. This was deemed unfeasible, as PhotoCube is a professional-to-expert level software that requires a somewhat deep understanding of the M3 data model powering it before it could be of benefit to any user. The time required to educate participants on this such that they could feasibly accomplish representative tasks presented to them without aid was determined to be unfeasible.

Due to these factors, the usability analysis was done with us as participants. This choice was considered optimal as we fulfilled some of the primary requirements for the intended product users: As part of our research, a theoretical familiarity with M3 had been established, yet there was still a lack of experience with PhotoCube. This would allow us to know what the system is capable of and supposed to facilitate, while maintaining the perspective of new users which could highlight usability issues the primary researchers had overlooked after their extensive interactions with the software. The method of exposing a "usability specialist" to a piece of software to dermine its usability is called usability heuristics. While it is often done, the "first law of usability" states that heuristic evaluations only have a 50% hit rate. Therefore, the expectation from this evaluation should follow with the assumption that a large portion of issues may be overlooked, and that a large portion of issues that are considered severe are not as such.

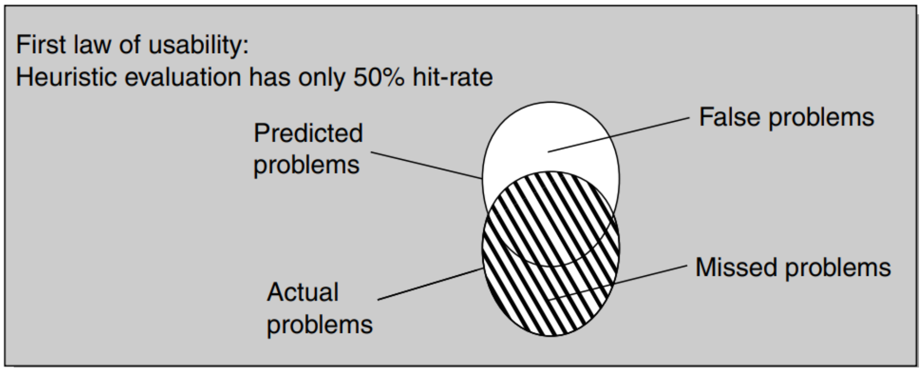


Figure 4: Usability heuristic evaluation [Lau05]

Conducting studies or experiments with oneself as participants can compromise data validity. To mitigate the effect of biases, it was decided to primarily focus the usability study on three things.

1. The number of operations required to complete baseline tasks
2. Unintended software behavior, bugs
3. Ease of navigating the UI

The perceived benefit of the first two points' focus is they are both replicable and (generally) constant for every user. The minimum number of operations required, such as for adding a hierarchy filter to an axis, would be the same for every user. Improvements to usability in that regard could then target lowering the operation count, improving the user's efficiency. Bugs that prohibit users from completing their tasks are also present irrespective of the user's perspective and were therefore found to be a good measure of potential usability improvements (depending on severity and frequency of the bug).

The ease of navigating the UI is the most "subjective" of the three foci and would be the most susceptible to compromised data validity in the form of subject bias. However, given the importance of a software's UI to its usability, focus on it was deemed essential. While this presents an issue, we attempt to account for it by relying on established design principles to diagnose UI issues such as the Gestalt Laws [Lau05] and via the post-improvement round of interviews, where the participants would also be exposed to the pre-improvement UI.

Practically, the usability tests were conducted by completing a set of tasks, such as completing a specific query, while scrutinizing every step. After a number of these tasks had been completed, some of the UI components were also tested in a manner similar to unit tests, specifically with the purpose of diagnosing inefficient design or buggy software.

It should be noted that while there are implicit differences between program defects and in-optimal design decisions, both can constitute usability issues and should therefore be considered equally when determining a software's usability [Lau05].

4.2 Findings

The data collection revealed a number of design deficiencies and software bugs that were then tested for reproducibility (where possible) and clearly defined. However, while the software's usability has several severe and moderate issues (which will be delved into), it did facilitate completion of all the tasks.

The most interesting findings of the usability inquiry are the issues with the highest severity. Issues with high severity are design decisions or technical faults that have been deemed to have a very high negative impact on the usability experience. More specifically by these issues being highly prevalent and manifesting in such a way that completing average-case tasks becomes more difficult.

Clear examples of these are a number of easily triggered software defects. A particularly notable example of this is the application resetting whenever the user presses the ENTER key while targeting an input field. Typing and searching via input fields is a key component of the application, and pressing ENTER to submit is an established norm of keyboard accessibility (similar to pressing TAB to navigate to the next element).[1] Instead, in the client, searches can only be executed by clicking the "search" button next to the input field. This issue is especially frustrating due to its ease of accidental triggering and its negative effect of completely resetting any queries the user had made.

An example of a severe design deficiency (rather than a technical one) is the dimension browser. The primary purpose of the dimension browser is to house the hierarchy filter browsing, the feature that allows for searching and adding hierarchies. A foundational principle of user design is to highlight what is important [Lun21]. The dimension browser, despite being a critical feature of the software, is by default minimized to the bottom of the screen. This issue is pronounced by the presence of the tag-filter on the left-dock. Tags and hierarchy filters serve distinct purposes but appear very similar. For example, both are searchable, both apply as filters, and both can yield very similar results when searching. However, tag filters cannot be applied to any axis. During completion of the first task, one of the participants couldn't locate the hierarchy filter, assumed it was the tag filter, and then thought their client was faulty when the behavior of the tag filters didn't match the expected results of the hierarchy filters. This issue is further compounded by a lack of responsive design, meaning the UI does not scale in size with higher resolution displays, leading to the

[1] https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide-chromium/accessibility/test-tab-enter-keys

dimension browser appearing even smaller on higher resolution displays.

Beyond technical issues, the primary usability issues of the PhotoCube client can be identified as being related to unclear UI, unexpected behavior, and low efficiency. Overall, this leads the PhotoCube client to have a number of severe and moderate usability issues that can coalesce to create a negative experience from a usability standpoint.

All of the issues identified have been categorized and collected into a schema for overview and to serve as a guide in the following section, where improvements to the client will be made. This schema can be found in the appendix A.1.

5 Usability Improvements

A considerable amount of moderate and severe usability concerns has been identified in the data visualisation component of the client - client code written in Three.js, responsible for the 3D browsing state. However, as a separate research project was investigating this in parallel with this project, it was deemed beneficial to exclude from this project's scope. Therefore, the focus for improvements was placed on the primary user interface elements and major bugs identified while exploring the usability of PhotoCube. These will be described in detail in the following section.

5.1 Implemented changes

Several of the usability issues outlined in the previous section have been successfully enhanced. The problems estimated as severe were targeted first, as they had the highest perceived negative impact on the usability of the PhotoCube client. These changes can be accessed in the usabiltiyFixes branch of the repository.[2]

As the first priority, the software defect that caused the entire application to reload after pressing "ENTER" in any input field has been resolved. This required small changes to two subcomponents: TagFilter (LD) and HierarchyFilter (DB). In both instances an event listener has been added to enable search functionality when the "ENTER" key is pressed.

A related problem was present in a situation where there was a need for clearing filters from the Date Filter (LD) or Dimensions (RD). If the user selected the "Select Year/Month/Day" option in the Date Filter or "Select filter" in Dimensions instead of clicking the "clear" button, a bug occurred which caused a syntax error leading to the application crashing and needing to be refreshed. This has been fixed by modifying the code of the "DateFilter" and "Dimension" components, such that "deselecting" a value has the same effect as clicking the

[2]https://github.com/biado/PhotoCube/tree/usabilityFixes

"clear" button.

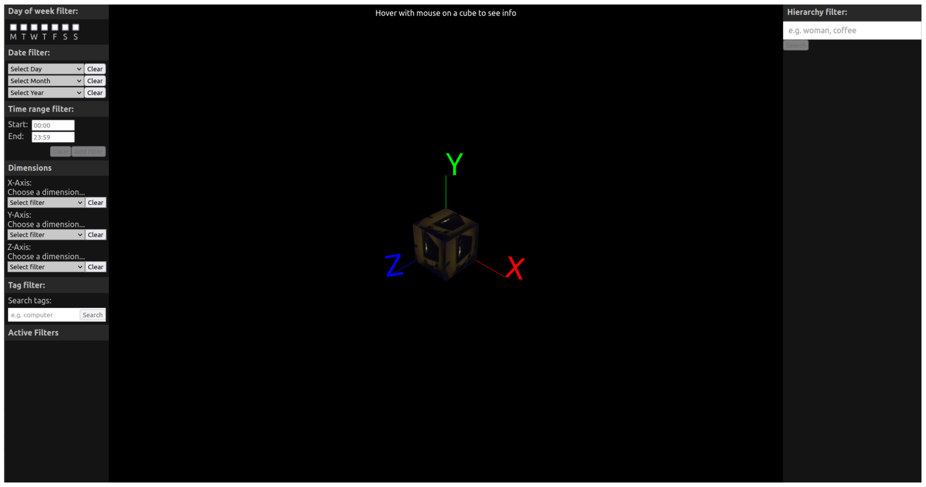
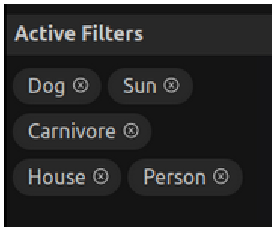


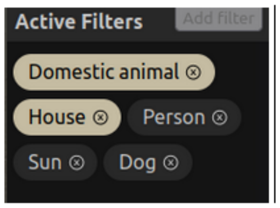
Figure 5: New PhotoCube UI implementation

Beyond rectifying software defects, a considerable change has been made to the layout of the PhotoCube UI. As described in the previous section, the placement of the Hierarchy Filter in the Dimension Browser caused several usability issues. Therefore the Dimension Browser has been discarded and the Hierarchy Filter has been moved to the Right Dock, where it is present in an expanded state by default. Furthermore, Active Filters and Dimensions have been moved to the Left Dock and the Browsing Mode has been completely removed, along with the Tagset Filter (Figure 5). The reason for complete removal of the last two components was that they were deemed not relevant in the context of this project. These changes are reflected in the client code, where the corresponding components have been moved or discarded accordingly. The new UI creates an environment easier to navigate, particularly for new users.

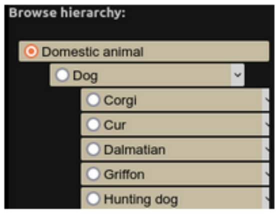
The last issue marked as severe that has been resolved is the way Active Filters are displayed. In the old state it was hard to differentiate between the Tag Filters and Hierarchy Filters. After they have been applied, all tags in Active Filters look alike. To make the difference clear, the background of hierarchies visible while searching the Hierarchy Filter has the same colour as its corresponding tag in the Active Filters (see Figure 6 for comparison). The design principle guiding this decision is that of Gestalt Laws in regards to contrast and color. Human psychology will implicitly connotate items of the same color with each other. Thus, a simple and efficient method to distinguish two types of elements from each other (that otherwise share many visual traits), is to give each type their own color or contrast [Lau05].



(a) Old Active Filters



(b) New Active Filters

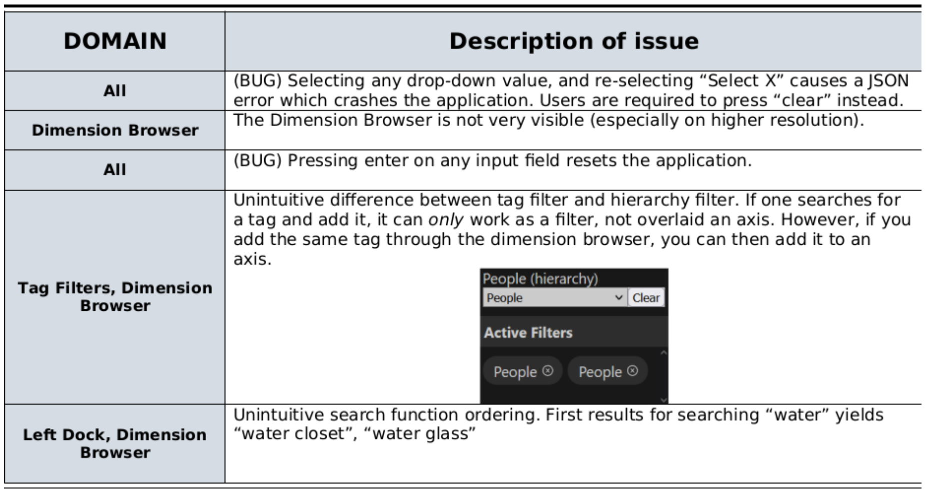


(c) New Hierarchy browser

Figure 6: Old vs new display of active filters

Lastly, a sorting function has been added to HierarchyBrowser and TagFilter components in the client code to process the resulting lists of tags before displaying them to the user. This fix addressed a usability issue where the search results were unordered and hence highly unintuitive. As a consequence, searching for the desired tag took more time than necessary, especially when the search results contained very long lists of tags. In the new version, the displayed search results are sorted alphabetically. This should increase search efficiency as users' expected results should be near the top rather than arbitrarily placed. The input "Car" will have "Cars" near the first result rather than "Car Wheel", for example.

Figure 7 presents a fragment of table with descriptions of usability issues which has been fixed and described in this section. A full schema of all identified issues can be found in Appendix A.1



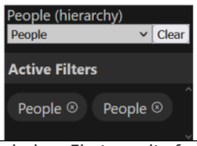
DOMAIN	Description of issue
All	(BUG) Selecting any drop-down value, and re-selecting "Select X" causes a JSON error which crashes the application. Users are required to press "clear" instead.
Dimension Browser	The Dimension Browser is not very visible (especially on higher resolution).
All	(BUG) Pressing enter on any input field resets the application.
Tag Filters, Dimension Browser	Unintuitive difference between tag filter and hierarchy filter. If one searches for a tag and add it, it can *only* work as a filter, not overlaid an axis. However, if you add the same tag through the dimension browser, you can then add it to an axis.  People (hierarchy) People Clear Active Filters People ⊗ People ⊗
Left Dock, Dimension Browser	Unintuitive search function ordering. First results for searching "water" yields "water closet", "water glass"

Figure 7: Resolved usability issues

5.2 Future work

Large parts of the identified usability issues have not been resolved. Decreasing the number of clicks required to project a Hierarchy Filter on an axis was the only "severe" issue not fixed. Potential solutions could include implementing a dynamic search, where results would start appearing as a user starts typing the query.

Among other important usability concerns which were not resolved is one related to arrow keys used for navigation. Pressing the arrow keys manipulates the dropdowns in addition to navigating between photos. This can lead to unwanted operations such as changing the data model or even crashing the application. Limiting the arrow keys to only work in the window with the photos displayed could be a potential solution to this issue.

Another noteworthy issue that should be worked on in the future is adding help sections to guide beginner users. Adding question marks with explanatory text pop-up next to each important part of the application would be an easy way to help new users become familiar with the system much faster. On a related note, the PhotoCube client can be navigated and modified by combining mouse clicks with holding keys (such as CTRL). However these features are not listed and the only way to figure them out is by trial and error. Including a short help section for application controls would be a natural way to eliminate that issue.

The described usability concerns have not been implemented mainly due to limited time resources and perceived lower feasibility of the tasks. As this research is a preliminary work with a main goal of becoming familiar with the application, many of them have also been deemed outside the scope of the project.

6 Evaluating changes to the PhotoCube client

6.1 Usability test - methodology

In this section, usability tests will be conducted to determine to what extent the changes to the client improved its usability.

To facilitate this, two external participants were chosen as software users. These participants were asked to engage in a simple usability test in which they were given a series of sequential tasks and asked to complete them. The choice for using external participants in these tests and not the initial usability evaluation is for a multitude of reasons. The primary reason is that it was deemed that the validity of the data would be compromised if we were to evaluate our own changes. Given that we only implemented changes we considered would improve the usability of the client, any data from a usability test on ourselves would be inherently biased.

However, the issue raised in the methodology considerations of the usability tests were still relevant. The participants, both of whom were master's students at ITU, would not be able to be representative of the target demographic. In an attempt to mitigate this, the tasks were designed to be more "atomic" than they usually would be. In practice, this manifests in what would likely be one task, e.g.

1. Create a query with 'Dog' and 'Water' on separate axes

into discrete tasks such as

1. Locate the hierarchy filter search
2. Add the 'Dog' and 'Water' hierarchy
3. Apply the 'Dog' hierarchy to the Z-axis
4. Apply the 'Water' hierarchy to the Y-axis

The reasoning behind this was that smaller and more explicit tasks would be easier to complete in a system that requires a lot of foreknowledge to use. An internal interview guide can be found in the appendix A.2.

The usability test followed a traditional think-aloud test format following Lauesen's guidelines [Lau05] One participant was given the tasks sequentially, one facilitator listened to the participant and asked as needed, and the log keeper noted any problems the participant encountered.

The participants were primed briefly on what the software was, an early representation of a multi-dimensional image browser, and that the system was in very early development, and that any failure of the participant to use the system or complete tasks was a fault of the system and not them.

Finally, all the tasks were set to be completed twice by each participant. First, all the tasks would be completed on one iteration of the software, after which the participant would then complete the same tasks on the other iteration. This was done with the purpose of creating an explicit comparison between the two versions. However, a natural pitfall to this is that a person who has already completed one set of tasks in one iteration, despite the versions' differences, would already be more familiar with the software and the second run through the tasks would likely be easier. As an effort to mitigate this, we ensured that one participant would start with the old version of the software and end with the new, with the opposite being the case for the other participant. While the familiarity would still be present at the second turn of the tasks, the different ordering would ensure that both iterations of the software would receive the benefit of this familiarity, so as not to skew data validity. Beyond the aforementioned concerns there are also potential issues of sample-selection and -size. It was not deemed feasible to mitigate these heavily, so this was not pursued. It is therefore understood that the findings of these usability tests are not to be seen as robust conclusions to the hypothetical of whether or not the changes improve usability, but rather an explorative study to catch major UX concerns and a preliminary indicator.

6.2 Usability test - findings

Generally, the findings of the usability test reinforced the hypothesis that the design changes and software defect fixes improved the overall usability of the application.

The report notes (found in appendix) list the tasks the participants were asked to accomplish, how the participant achieved this, notable remarks during the experience, and (if present) what kind of fault there was.

The most notable issue was present with the dimension browser in the old iteration of the client. As remarked in the initial usability evaluation section, the dimension browser constitutes a critical feature of the client yet by default it was minimized to the bottom of the display. This issue was so severe, that both the participants failed to complete the first step of the first task (locating the hierarchy browser and entering a search) without aid from the facilitator. This constitutes a severe lack of fit for use, and (naturally) also severely diminished the subjective satisfaction. This issue was not experienced by either of the participants with the new iteration. With the new iteration, both participants quickly identified the hierarchy browser on the right dock and used it.

The addition of the hierarchy browser to the right dock did introduce new, unforeseen issues. On screens with lower resolution, the width of the right dock did not fully encompass all the entries in a search. When opening children and grandchildren in the tree-like hierarchy view, the child-entries would be offset slightly to the right (as is intended, to indicate their parent). With the lower width, this meant that the children would ultimately be cut off. As can be seen in Figure 8 where the “show children” buttons for children of Person are cut off. The window can be scrolled horizontally, but this is not optimal.

The ENTER key restart bug was also highly present and diminishing of the users’ satisfaction. Both participants discovered it fairly quickly and modified their use thereafter, but noted that it was contrary to their expectations. As an example of usability adaptation, the participant who used the old iteration first and experienced this issue, did not attempt to launch searches with the enter key without prompt when shown the second iteration, expecting the behavior to replicate.

Another significant issue was how the search results were ordered in the old

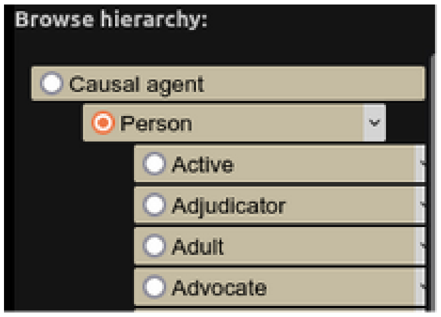


Figure 8: Hierarchy browser in new implementation

iteration. Both participants spent an increased amount of time and exclaimed minor annoyance at the ordering of the search results in the old client. Their perception of the issue was that it was one of broken expectations.

When having completed several tasks, distinguishing between tags and hierarchy filters became an issue on the old client. At a certain task, the participants were asked to clear only the active tag (direct) filters and not the hierarchy (projected) filters. Without full memory of which were which, neither could complete the task on the old client.

There were, however, changes made between the clients that the participants didn't notice or did not impact their completion of the tasks. For example, the bug-fix that prevented a crash when resetting filters without pressing the "clear" button was not triggered by either user. As per our usability schema, this issue was listed as severe, but these tests indicate that, despite the destructive potential of the issue (full app restart), it would perhaps not be common enough so as to warrant its severity rating.

Overall, it can be concluded from the usability study that the most important changes are related to design decisions that deal with ease of use and clarity. With the new design decisions, efficiency of the users increased as they completed their tasks quicker. Using the Gestalt Law of contrast on similar types also served to distinguish their elements from each other such that the participants gained clarity on which element belongs to which type [Lau05]. Of the bug-fixes, only one (ENTER to search) had an impact on the participants' experiences. This is despite the other bugs being listed as SEVERE, indicating a high likelihood of encounter during regular use. While the bug-fixes reduce the potential for crashes, their lack of presence during the usability test indicates that the severity could have been reduced, and focus on fixing other issues could have been had. However, this is not necessarily a condemnation of the process itself.

In defense of the old client, the left-dock, right-dock layout that the new client continued proved to efficiently communicate the important elements for interaction. With only a basic verbal introduction to the data model and software, users quickly understood that the section in the middle was for browsing, and that controls to manipulate this would be in the edges of the screen. The only major issue found with this UI was the dimension browser at the bottom.

7 Conclusion

In this paper, the M^3 data model and the software built on it were introduced. The software, PhotoCube, was subject to a usability inquiry the result of which was a schema that categorized and prioritized usability issues of the software. This was used as a guide to prioritize implementation changes to the client. Using UX design principles, new solutions were designed and implemented for the client. After the implementation phase, a minor usability test with two external participants was conducted.

Due to methodological and statistical concerns it cannot be surely concluded that the usability of the PhotoCube client improved with the design changes and bug-fixes the new implementations brought, however the final usability tests do indicate that this could be the case. For now, the client is still being developed by a team of researchers who can hopefully use the design implications if not concrete implementations developed during the course of this project.

The secondary purpose of this project was to serve as a foundation for a thesis on the M^3 model in virtual reality. Creating a PhotoCube-like application in VR would introduce scalable multimedia analytics to an entirely new medium. While not everything from this project is directly translatable to the VR medium, the foundation of the application, the data model, remains largely the same.

References

[Cla19] Peter Øvergård Clausen. “A Web Browser Based Implementation of PhotoCube”. MA thesis. IT University of Copenhagen, 2019 (cit. on p. 3).

[Lau05] Søren Lauesen. *User Interface Design: A Software Engineering Perspective.* Pearson Addison Wesley, 2005 (cit. on pp. 9, 10, 12, 15, 17).

[Lun21] Mircea Lungu. *Graphical Principles for Web Design: Hierarchy, Layout, Color Typography.* ITU, Technical Interaction Design. 2021 (cit. on p. 10).

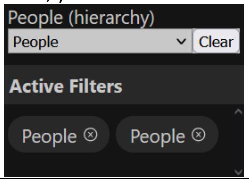
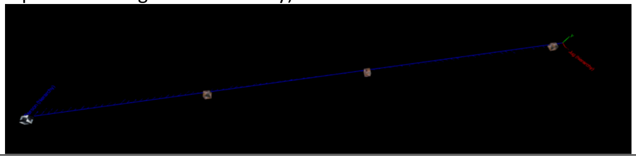
[Sig11] Hlynur Sigurórsson. “PhotoCube: A Multi-Dimensional Image Browser”. MA thesis. Reykjavík University, 2011 (cit. on p. 3).

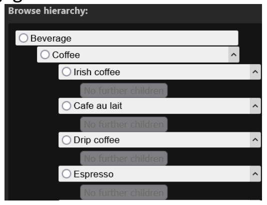
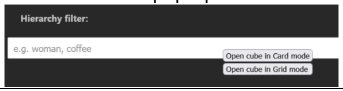
[Tóm11] Grímur Tómasson. “ObjectCube – A Generic Multi-Dimensional Model for Media Browsing”. MA thesis. Reykjavík University, 2011 (cit. on p. 3).

A Appendix

A.1 Usability schema

- **Domain**
  - The primary domain of the usability issue (React docks)
- **Estimated severity**
  - **SEVERE** - Issues that severely impact the usability experience and are difficult to avoid during regular use
  - **MODERATE** - Issues that moderately impact the usability experience and/or are not often encountered during regular use
  - **MINOR** - Issues that have little impact on the usability experience and/or are rarely encountered during regular use
- Estimated solution feasibility
  - **HIGH** - Proposed solution should be implemented with few minor changes/additions
  - **MEDIUM** - Proposed solution would require a moderate amount of work and/or refactoring existing code
  - **LOW** - Proposed solution would require large amounts of work and/or refactoring existing code

DOMAIN	Severity	Solution Feasibility	Description of issue	Proposed design change
All	SEVERE	HIGH	(BUG) Selecting any drop-down value, and re-selecting “Select X” causes a JSON error which crashes the application. Users are required to press “clear” instead.	Rectify the software defect. Selecting drop-down options should not crash the application.
Dimension Browser	SEVERE	HIGH	The Dimension Browser is not very visible (especially on higher resolution).	Move the dimension browser from the bottom dock or increase its or change its color to highlight importance.
All	SEVERE	HIGH	(BUG) Pressing enter on any input field resets the application.	Rectify the software defect. Pressing ENTER should not crash the application.
Dimension Browser	SEVERE	MEDIUM	Adding hierarchies takes many operations 1. open dimension browser 2. search filter 3. press search 4. select specific hierarchy (unintuitive result ordering) 5. select specific (sub)hierarchy 6. press “add filter” 7. indicate which axis filter applies to	Include enter-to-search Intuitive search ordering (sort by length, alphabet) Ability to add filter to axis directly from D-browser Make search-box a combo-box These would decrease the operations required to add a hierarchy, making it more efficient.
Tag Filters, Dimension Browser	SEVERE	MEDIUM	Unintuitive difference between tag filter and hierarchy filter. If one searches for a tag and add it, it can *only* work as a filter, not overlaid an axis. However, if you add the same tag through the dimension browser, you can then add it to an axis.  People (hierarchy) People Clear Active Filters People ⊗ People ⊗	Visual distinction between tag filters and hierarchy filters. Law of contrast.
Left Dock, Dimension Browser	MODERATE	HIGH	Unintuitive search function ordering. First results for searching “water” yields “water closet”, “water glass”	Sort output of search by length of matching strings in ascending order such that the first terms are the ones that are closest in length to the search term. Thereafter, alphabeticlly.
Grid	MODERATE	MEDIUM	The order of the photos’ appearance in the grid browser is not explicitly stated anywhere	Show the sorting method for grid-based viewing.
All	MODERATE	LOW	Arrow keys are used for navigation. However, they can also manipulate dropdowns if the user has just pressed them. E.g., adding the “Woman” tag from the dropdown, and then using the arrow keys to navigate the model. Pressing the UP key would also manipulate the drop-down to select the above option, changing the data model or crashing the application if it reverts to default value.	Force arrow key interaction to be solely focused on the data model window, or alternatively, explicitly disable arrow key manipulation of all inputs.
Cube browser	MODERATE	LOW	Empty sub-hierarchies still appear in the cube-browser. This can make for difficult browsing if you’re specifically just looking for, e.g., “Person” and “Jug”. Jug has one sub-hierarchy and Person has *many* (figure: blue line represents Person sub-hierarchies; the cubes are pictures fitting a sub-hierarchy). 	Allow for hiding/collapsing sub-hierarchies in the cube-viewer if they’re empty.
All	MINOR	HIGH	Application has no help-section or beginner’s guide.	Add menu option with explanatory text.

All	**MINOR**	**HIGH**	Application features controls and modifiers (e.g. holding CTRL when browsing), but these are listed nowhere. The user must intuit them.	Include short guide/help section for controls.
Dimension Browser	**MINOR**	**HIGH**	The dimension browser can only be expanded by pressing the small icon on the right-side. This design is unintuitive and does not scale well with larger images.	Allowing to expand the dimension browser by clicking anywhere on the bar.
Dimension Browser	**MINOR**	**HIGH**	Must use a minimum of three clicks to add tag-set filters (from the dimension browser).	Changing tag-set filters from drop-down to checkboxes?
Tag filter	**MINOR**	**HIGH**	Tag set filter has both “day of week string” and “day of week number”. What is the purpose of two different methods of selecting the same thing?	Merge into one tag, allow user to choose which they prefer.
Tag filter	**MINOR**	**HIGH**	Must press “add filter” after adding a tag, unnecessary operation.	Automatically add filter after selecting it
Card Browser	**MINOR**	**MEDIUM**	Browsing in Card based viewing is only possible via arrow-keys. Not in line with rest of the application that is mouse-navigable.	Show arrows on left and right side of images in card-based browser that allow for navigation using mouse.
Dimension Browser	**MINOR**	**MEDIUM**	Browser hierarchy shows dropdowns for entries with no further children, this makes exploration of the hierarchy difficult, as the user doesn’t know which hierarchies are “leaves” or not until they engage with them.  Browse hierarchy: Beverage Coffee Irish coffee No further children Cafe au lait No further children Drip coffee No further children Espresso No further children	Remove down-arrow for “leaf” entries.
Dimension Browser	**MINOR**	**MEDIUM**	Can only add one filter at a time from the hierarchy filter. if one wants to add, e.g., several but not all types of alcohol (N number), it will take N*2 clicks in the best-case scenario	Allow selecting multiple filters at once and adding them all when pressing “add filter”.
All	**MINOR**	**MEDIUM**	Right clicking on the model viewer to show the “open cube in x” pop-up and then opening the dimension viewer retains the pop-up.  Hierarchy filter: e.g. woman, coffee Open cube in Card mode Open cube in Grid mode	Clicking anywhere in the app should remove the pop-up
Grid Browser	**MINOR**	**LOW**	Can’t full-screen individual images from grid-mode	Double-clicking an image in grid-mode should open the image in card-viewer, similar behavior to most conventional file browsers.

A.2 Interview Guide for testing the usability of the Web-Based implementation of PhotoCube

This interview guide is only for the researchers to align procedure and expectations prior and during to the interviews. This interview guide is not to be shown to the participants.

Participants

Both participants are master's students at ITU. Under ideal circumstances, the tests would feature a random selection of people who are representative of the envisioned user-group of the software. However, despite that it is determined that the interviewees will still be able to provide valuable feedback in regard to the usability, given that they are not familiar with the system and the changes whatsoever.

Structure

The interviews are going to be think-aloud tests. Think-aloud tests are a common tool to establish the usability of a system. Essentially, think-aloud tests prompt the participants to use a piece of software while explicitly verbalizing their thoughts. This is a more formal definition:

Definition: *In a thinking aloud test, you ask test participants to use the system while continuously thinking out loud — that is, simply verbalizing their thoughts as they move through the user interface.*

The participants will be given a set of narrow tasks that they have to complete both in the initial PhotoCube implementation, and in the new version. In other words, each task must be completed twice, once in each implementation. This is done so the participant can verbalize the difference they perceive between the two – which one they perceive as more user friendly.

Tasks

For the think-aloud test, tasks must be presented. Because the participants are not representative of the actual users, and because they are not familiar with the data model, the tasks will be very narrow in scope. The normal users of this kind of tool would likely be professionals familiar with the data model, as the participants to do not fulfill these criteria, the testing will focus on narrower task that can still give good data as they are more "universal" usability characteristics (e.g. how a search function should react to the enter-key).

Task 1 – DIMENSION BROWSER
LOCATION: FROM THE START OF THE APPLICATION
Find the hierarchy filter, search for "Car", Add "Car(3284)" as a filter

Task 2 – FILTERING
LOCATION: JUST FOLLOWING TASK 1

Add the car filter to the Z-axis

Task 3 – DIMENSION BROWSER
LOCATION: JUST FOLLOWING TASK 2
Find the hierarchy filter, search for “liquid”, from liquid – choose “liquid:substance”, add “alcohol” as a filter

Task 4 – FILTERING
LOCATION: JUST FOLLOWING TASK 3
Add Alcohol to the Y-axis

Task 4 – FILTERING
LOCATION: JUST FOLLOWING TASK 4
Clear the axes and remove all active filters

Task 5 – FILTERS
LOCATION: IN THE DIMENSION BROWSER HIERARCHY FILTER AND TAG FILTER
Add two hierarchy filters: “animal”, “human” and two tag filters: “dog”, “child(9229)”

Task 6 – HIERARCHIES AND TAGS
LOCATION: FOLLOWING TASK 5
Remove all tag filters

Task 7 – FILTERING
LOCATION: FOLLOWING TASK 6
Add Date Filter: Select year 2018. Add time range filter 12:00 to 16:00. Reset all the filters.

Task 8 – SEARCHING FILTERS
LOCATION: FOLLOWING TASK 7
In the hierarchy filter search for ”person”. Add ”person: casual agent”, find and add ”traveler” filter.