

Documentation - IR Master Assignment

Introduction - Usability engineering for this project

The usability engineering process for this project can be divided into the following steps:

- Requirement Analysis - what does the interface need to contain and in which way should it be designed in order to be useful
- Mockup Generation - design sketches of the interface based on the requirements of the project
- Feedback Evaluation and resulting design - incorporate the feedback on the mockups into the design and create a draft that combines the strengths of the mockups and reduces their weaknesses

This document is structured along these steps from the design process.

Requirement Analysis

Since this project is quite tightly defined by the task, requirement analysis as practiced in guerilla usability engineering will be implemented. This means instead of a large statistics-driven and time consuming approach we will infer the requirements of this project from the task and the use case of the application.

Requirements inferred by functionality

1. User needs to be able to input a query of his choosing for the search
2. The results of the search need to be displayed to the user in an orderly fashion
3. Given that only 10 results should be returned for each query and assuming a normal screen resolution, actually 'navigating' the result list , i.e. changing pages, should not be required

In retrospect it has to be said that the function to choose between using an existing index or creating a new one were unintentionally not included in the requirements and the mockups resulting from them. This function later found it's way into the final version.

Requirements inferred by usability guidelines & best practices

1. Design should be driven by function. Core of the application is the search and the interaction with the results provided by the program. UI should promote these core

functions and support them accordingly. (Dieter Rams: “Good Design Makes a Product Useful”)

2. Design should be responsive. User needs feedback as to what's happening - if it takes some seconds to display search results, tell the user so. (Jakob Nielsen: “Visibility of system status”)
3. In reference to the first point listed here: good design is as little design as possible. The user needs to input a query, so we need a text entry field. The user needs to enter/confirm this query and start search, so according to best practices we need a 'search' button. Other than that, we need to display the result list in a space intended for this. Everything beyond these elements is to be seen as an extension, and should be well justified. (Dieter Rams: “Good Design Is as Little Design as Possible”)

Mockups

Mockup 1:

The idea behind this mockup is to combine the requirements listed above in a way that is as functionality driven as possible.

The text entry field that is used to enter the query has a so called ‘hint text’ that gives the user a hint on what input is expected in the field. This makes a separate text label to designate the input field as such obsolete.

Although hitting enter in the search field might be a sufficient means to start the search we still included a button that starts the search in order to foster functionality understanding by the user. A single text entry field might confuse the user, while a text entry field in combination with a search button reflects the common expectation of a search mechanic.

The results are listed below in a separate frame of the interface which would be blank before the first search. The single results are visualized as separate elements in the frame in order to allow a better visual differentiation between them. The elements contain the rank, title and url on the top level and the summary of the result in the lower level. For longer titles this might be a problem, so the arrangement is not fixed here.

CodeSearch

Enter query here Search

Rank - Result title <http://yoururlicouldbehere.com>
Summary dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.

Rank - Result title <http://yoururlicouldbehere.com>
Summary dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.

Rank - Result title <http://yoururlicouldbehere.com>
Summary dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.

Rank - Result title <http://yoururlicouldbehere.com>
Summary dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.

Rank - Result title <http://yoururlicouldbehere.com>
Summary dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.
Donec dolor ex, sodales vel nibh vel, pellentesque feugiat mauris.

Mockup 2:

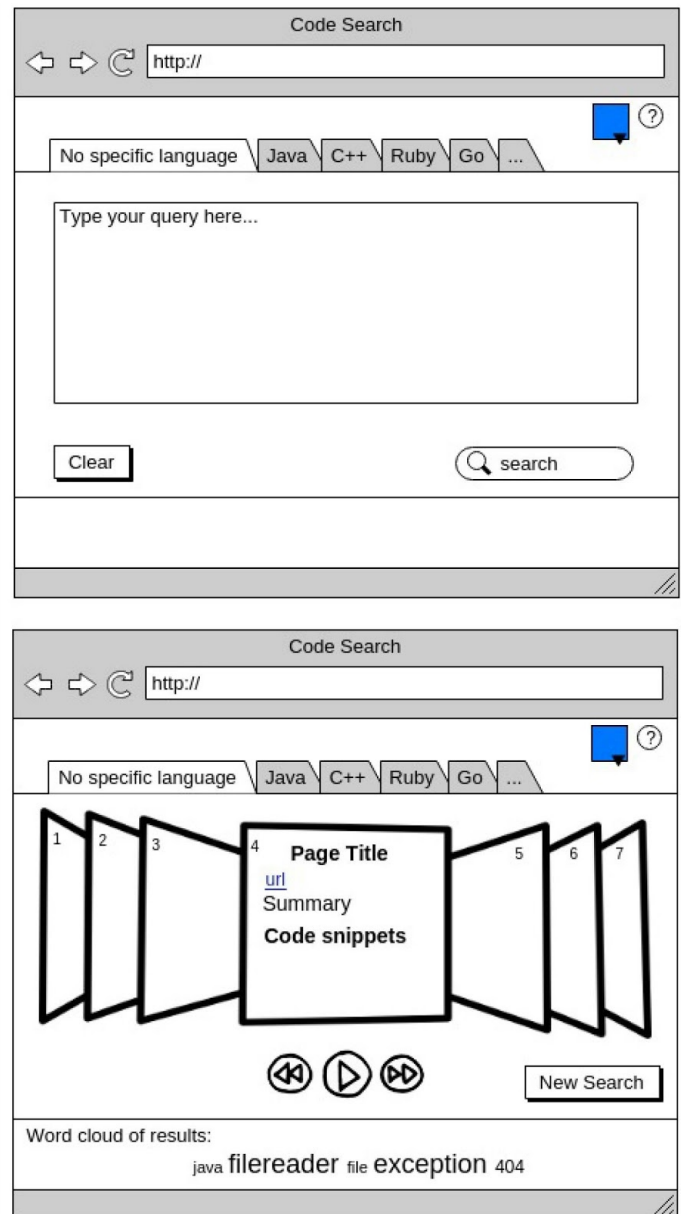
In this mockup some different concepts were tested. The interface is divided into two screens - the first (upper part of the image) is used to create a search query and the second one (lower part) is used to browse the results of the search.

The page to input the query is divided into several tabs for different programming languages with a separate tab for searching in no specific language. The entry field has more space assigned to it than in mockup 1 and also contains a hint text.

Two buttons on the bottom of the page are placed in the left and right corner with the left one to clear the input in the entry field and the right one to start the search.

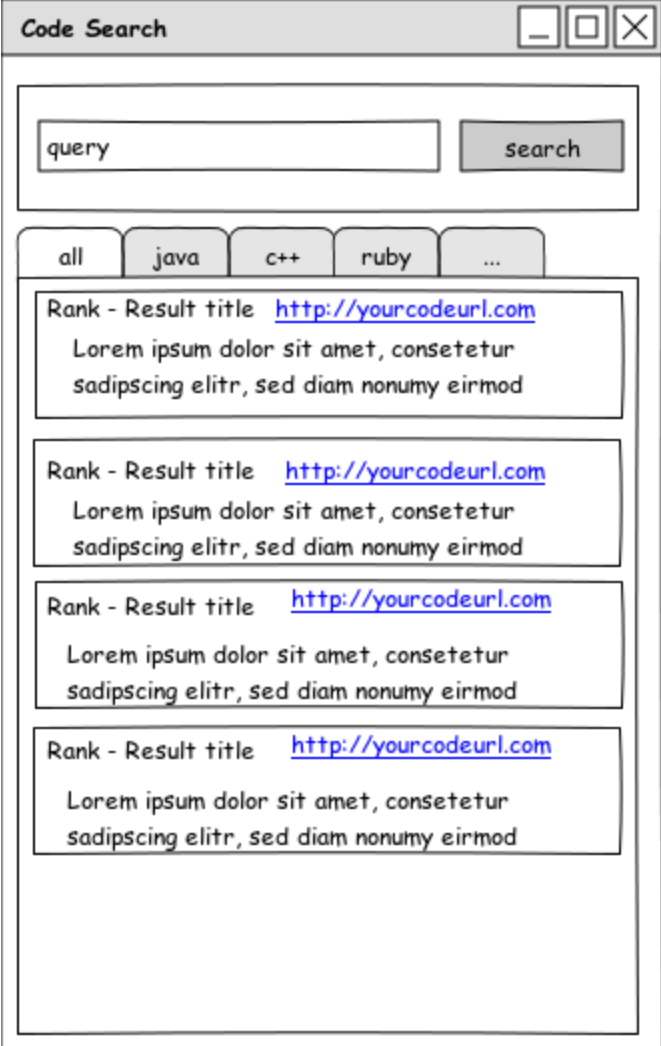
The elements of the result list are ordered horizontally starting with the best result on the left going to the worst result to the right. The intention here was to create a method of display that is visually more appealing than a simple list and has a more prominent display of the currently selected result. The results are browsed by using navigation buttons below the results as seen in the picture. To get back to the first page to be able to type in another query a button 'New Search' is located in the bottom right of the result frame.

Below the result frame is a separate frame displaying a horizontal word cloud of the most frequent terms for all 10 results. This is intended to give a quick overview of the content of the results.



Mockup 3:

This mockup should incorporate the idea of dividing the search into tabs for different programming languages. Other than that, the mockup is the same as mockup 1.



The mockup shows a web application titled "Code Search". It features a search bar with the placeholder text "query" and a "search" button. Below the search bar are five tabs: "all", "java", "c++", "ruby", and "...". The "all" tab is currently selected. The search results are displayed in a list of four items. Each item consists of a "Rank - Result title" followed by a blue hyperlink "http://yourcodeurl.com" and a block of Lorem Ipsum text: "Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod".

Code Search

query search

all java c++ ruby ...

Rank - Result title <http://yourcodeurl.com>
Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod

Rank - Result title <http://yourcodeurl.com>
Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod

Rank - Result title <http://yourcodeurl.com>
Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod

Rank - Result title <http://yourcodeurl.com>
Lorem ipsum dolor sit amet, consetetur
sadipscing elitr, sed diam nonumy eirmod

Evaluation via User Tests

Test procedure and testers

In order to evaluate the previously presented mockups, a short user test was performed with 4 testers total. The tests were carried out using Skype for communication between the tester and the developer and followed the following procedure:

1. The developer started the skype call and greeted the tester.
2. Then the tester was introduced to the context of the application and the test by a statement prepared prior to the test.
3. One of the three mockups that the developer chose arbitrarily was shown to the tester by sending him a link to the mockup image.
4. The tester was asked to have a look at the mockup and to communicate his thoughts about the mockup in a 'think-out-loud'-fashion.
5. The developer made rough notes of the feedback from the tester.
6. Steps 3-5 were repeated for the other two mockups.

All the testers were familiar with at least one programming language and are friends of one of the developers - a fact that might bias the results to some extent. More info of the testers is shown in the table below.

Tester No.	Age	Sex	Occupation
1	26	Male	Master Student (Computer Science)
2	24	Male	Master Student (Media Information Technology)
3	24	Male	Bachelor Student (Computer Science)
4	31	Male	Software Developer

User Feedback

The feedback the users gave regarding the different mockups is listed in the table below. The number of the mockups in the first column are the same as in this document. Similar feedback from multiple testers regarding the same mockup was combined, indicated by multiple tester numbers in the respective cell.

Mockup No.	Tester No.	Positiv / Negativ / Neutral (+/-/*)	Feedback
1	1,4	*	Expressed wonder why the project is a regular program rather than a website. It was explained to him that this is a project requirement.
1	1,3	-	thinks the GUI looks visually 'boring'. Reasons given for that were "lack of color" and "sterility"
1	1,2,3,4	+	functionality is clear
1	1,2,3,4	+	Entry field and search button meet user expectation from known search engines
1	1,2	*	missing scrollbar - fault in the mockup, depends on window size
2	1,3	-	searchbox bigger than needed (quote tester 3 "What am I to right there, a whole paragraph?")
2	1,2,3,4	-	Questions the concept of tabs for different languages. Used to specifying the language in the query.
2	1,3,4	+	functionality is clear
2	4	- / *	confused 'search' button in the bottom right of the first screen with an additional text entry field.
2	2,3	-	'clear' button is unnecessary
2	1,2,3,4	-	Result list design unfitting, little overview. Quote tester 4 "looks more like a playlist than a resultlist"
2	1,3,4	-	Query not visible from result list
2	1,3,4	-	Related to the critique on the tab design: tab for 'no specific language' seemed pointless
3	1,3	-	same as for mockup 1 - GUI looks visually boring
3	1,2,3,4	+	As for Mockup 1: entryfield and search button fit user expectation
3	1,2,3,4	-	tabs better than in mockup 2 but still seen as unnecessary / overly complicated

3	1,2,3,4	+	functionality is clear
3	1,2,4	*	missing scrollbar - fault in the mockup, depends on window size

Summary of the received feedback & it's implications

Overall the user test helped a lot to identify weaknesses in the proposed designed and evaluate different approaches. All three mockups managed to illustrate the functionality of the program in an understandable manner, although the tab designs of mockup 2 & 3 and also the horizontal result list of mockup 2 were regarded as more cumbersome than useful.

Due to little visual variety the mockups 1 and 3 were seen as 'boring', the lack of color was a big factor in this. Because the mockups were intended to test the use of different UI elements and arrangements, this is a point which was clear from the beginning.

In general the user test showed that catering to the user expectations regarding the appearance of the searchfield, searchbutton and result list helped communicating the functionality of the program since the users associated this layout with search functionality.

These insights were used in the process of refining the gui for the final implementation. The general structure of mockup 1 would be kept and be the base for the implementation. In order to accommodate for the settings section of the program which forgotten about when doing the design mockups it was decided that a seperate tab will provide both sufficient space for this and also offer the possibility to keep the section discrete until the user actually needs to perform changes in the settings.

The changes in the GUI will be shown in the part below addressing the actually implemented GUI.

Final GUI layout and implementation

The results from the user testing were incorporated into the first mockup to act as the base for the final design. This will be presented in detail below.

General Layout

The GUI is divided into three tabs that keep the content of the application stored in a way that allows for both efficient use of space and an easy overview of the application.

The tab approach was also present in the second and third mockup but their usage was seen as unnecessary. Since the settings feature was overseen in the mockup design, tabs were considered a good solution to include the settings into the program.

The Search Tab

The search tab is the first of the three tabs and is selected by default. Here the layout as seen in mockup 1 is realized with only small changes.

The query input field is at the top of the tab content and is long enough to accommodate most queries. Next to it is the 'Search' button that starts the search.

Below that the result list is shown.

Here the rank, title, url and summary of the page are displayed together with the relevance score. To improve fast evaluation of the search item by the user, terms that are part of the query are highlighted in the summary of the result.

Additionally, if the document has code, it's first snippets are displayed when the mouse hovers over the url.

Since users in the text disliked the idea of separate tabs for different programming languages the specification of the language the results should be related to is done as part of the query as the testers would have expected in the first place.



We also added some functionality that wasn't provided by the standard JSwing objects: the URLs were made clickable, a simple menu allowing right-click copy was included in the results list, and another menu allowing to cut, copy and paste was added to the search box. Additionally, the undo and redo functions were provided for the search box.

The Settings Tab

In order to allow the user to adjust the program to his liking the 'Settings' tab provides several functionalities.

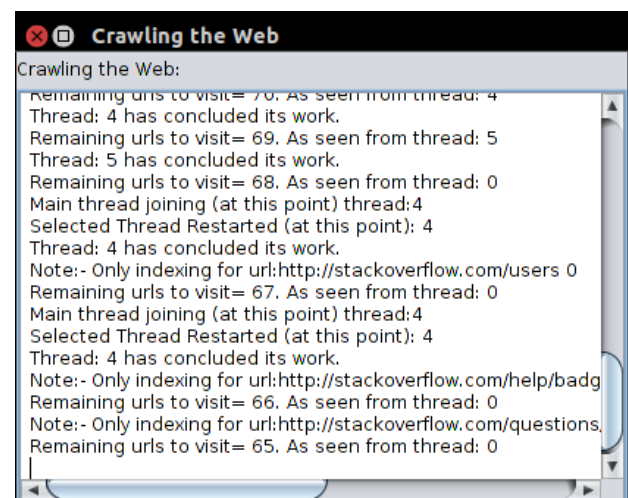
By default - meaning without any setup from the users side - an index is used for the search that is shipped with the program. By pressing the 'Use default index' button the program goes back to using the original index. On the other side a user can select an already created index with the 'Select another index'. The currently selected index is being shown at the top of the settings tab in form of a filesystem path.

Below that on the left and right side respectively are buttons to show the pages that were visited by the crawler and the pages that were excluded from the crawling process.

In the lower part of the tab the settings for creating a new index are located.

The user can set a crawling depth for the crawling process and can pass a list of URLs that are supposed to be used as seeds for the crawler. By clicking the 'Submit' button, the crawling process is started.

The crawling process opens up a separate window that will display status information to the user to keep him in the loop about what is happening. Even though he might not be interested in the mostly technical output that is printed there, it is important that the user can visually notice that the program is currently doing something - a simple 'Crawling the web, please wait'-message would not have the same effect.



To prevent inconsistencies in the index and possible program anomalies caused by them, it is not possible to close this window until the crawling has finished. Although it violates Nielsen's "User Control & Freedom" heuristic, the benefits of this with regard to Nielsen's "Error Prevention" heuristic justify this decision.

As we did in the Search tab, we also found the need to add extra functionality not provided by the standard JSwing objects: a simple menu allowing right-click copy was included in the windows for Crawling the web and for displaying the lists of pages. Another menu allowing to cut, copy and paste was added to the text area where the seeds can be introduced. The undo and redo functions were provided for this text area as well.

One additional factor that was considered w.r.t error prevention was to verify with users before crawling for depths higher than 5.

