Introducing Usability Practices to OSS: The Insiders' Experience

Stanisław Osiński[1] and Dawid Weiss[2]

[1] Poznan Supercomputing and Networking Center,
stanislaw.osinski@man.poznan.pl
[2] Institute of Computing Science, Poznan University of Technology,
dawid.weiss@cs.put.poznan.pl

Abstract. This paper presents a case study of introducing usability practices to a small open source project called Carrot^2. We describe our experiences from a point of view of an active Carrot^2 developer, who is at the same time a usability enthusiast and practitioner. We perform a critical analysis of the system's original user interface and describe the steps we took to improve it. We also analyse the success factors and the impact of the whole redesign process.

1 Introduction

The growing reliability of open source software (OSS) has led to widespread adoption of OSS server-side packages, such as the Apache web server or MySQL. At the same time, debates continued on whether OSS can also effectively replace proprietary end-user applications [1]. One important factor determining the success of the latter is the quality of their user interfaces (UIs). As opposed to traditional OSS users, who like taking on technical challenges, typical non-experts will simply want to complete their tasks quickly, minimising the effort required to learn and operate the software. Consequently, while the low quality of UI is unlikely to discourage experts from using OSS, it can potentially prevent massive adoption among end-users. Thus, problems of *usability*, i.e. the ease of learning and efficiency of using, of OSS are becoming increasingly important.

This paper is a case study of introducing usability practices to a small open source project called Carrot^2. We describe our experiences from a point of view of an active Carrot^2 developer, who is at the same time a usability enthusiast and practitioner.

2 Improving the user interface of Carrot^2

The aim of the Carrot^2 project is to provide a flexible framework for building *search results clustering* engines. Search results clustering is a web mining technique that simplifies browsing of search results by dynamically organising them into thematic folders (Fig. 1(d)). In response to the query 'tiger', for example,

the user would be presented with search results divided into such topics as 'Mac OS', 'Tiger Woods', 'Security Tool' or 'Tiger Attack Helicopter'.

Carrot^2 started in 2002 with a primary goal to enable rapid experiments with novel web mining techniques [2]. But as the project matured, we decided to place greater emphasis on supporting end-users, which required making Carrot^2's web-based interface easier to learn and use. We divided the process of redesigning the UI into four major stages: evaluation of the original design, prototyping of the new design, usability testing, implementation and evaluation of the redesigned version, all of which we describe below.

2.1 Evaluation of the original design

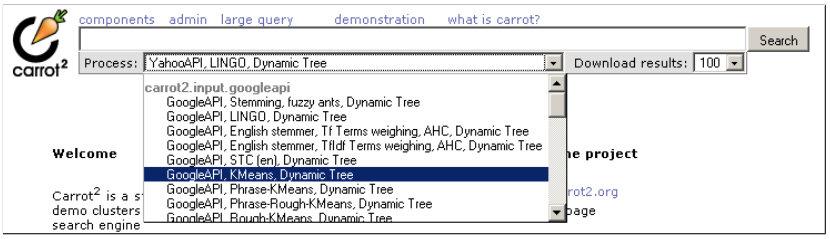
The original design of the user interface was aligned with the system's primary goal at that time – attract text mining researchers – and it perfectly served its purpose. After shifting the emphasis to end-users, however, some elements of the UI would not be appropriate anymore. Below we summarise the most important problems that we planned to address.

Choosing sources of search results. The original UI combined two different aspects – choosing the source of search results and the clustering algorithm – into one *Process* combo-box (Figure 1(a)). One problem with this approach was that these two choices are independent, and the original UI made it impossible to use certain combinations of search sources and clustering algorithms. A more serious problem, however, was that many end-users would care less about changing the clustering algorithm than the search source. Unfortunately, the original UI would force them to choose e.g. between: *'GoogleAPI, English stemmer, Tf Terms weighing, AHC, Dynamic Tree'* and *'YahooAPI, LINGO, Dynamic Tree'*, both of which are meaningless for a non-expert.

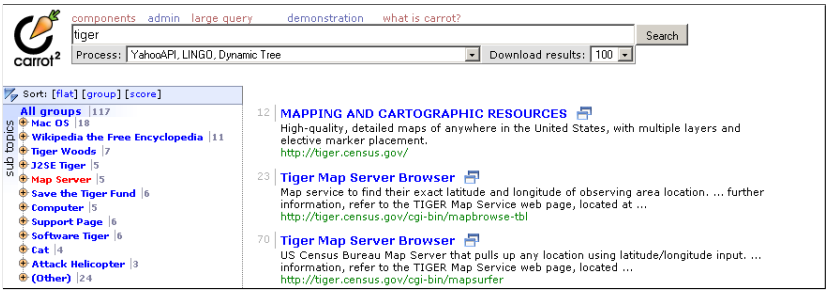
Choosing search results sorting order. The presentation of search results and thematic folders in the original UI was based on a variant of the two-panel selector pattern, in which the folders were shown on the left-hand side of the screen and the corresponding search results appeared on the right (Fig. 1(b)). One element of this interface that might be confusing to non-expert users was the choice of results sorting method located above the folder list. Not only was it put in the inappropriate visual context (sorting order referred to search results shown on the right and not to the folder list), but also it was not clear what different sorting orders meant and which of them was currently selected.

Technical jargon. At a few places the original UI used technical and specialised vocabulary not easily understood by non-experts. One example is labeling the combo box for choosing the search results source (Yahoo!, Google, MSN) as *Process*[1]. The combo box itself contained hard to understand items such as: *'Carrot^2.input.snippet-reader.alltheweb'* (internal identifier of a search results source) or *'Rough-KMeans'* (name of a text clustering technique).

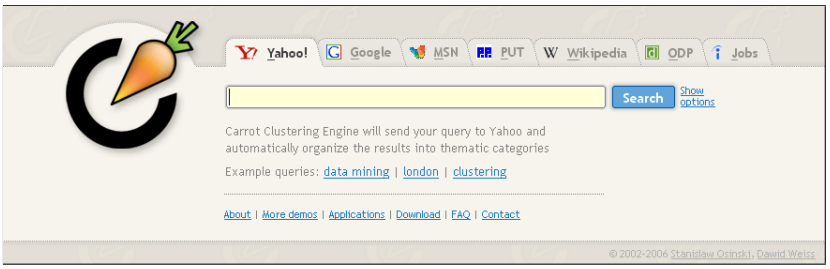
[1] The internal architecture of Carrot^2 is based on pipelines called *processes*.



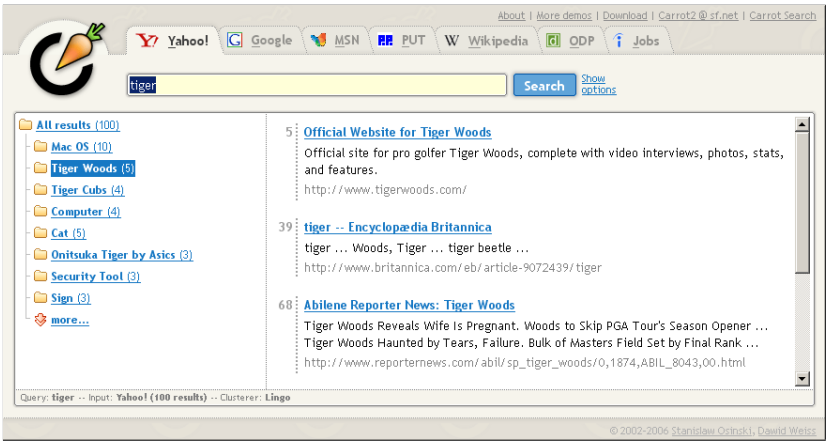
(a) Startup screen of the original user interface.



(b) Results page of the original user interface.



(c) Startup page of the new design.



(d) Results page of the new design.

Fig. 1. Carrot^2 user interface screenshots, available on-line at http://www.carrot2.org.

The good points. Despite the deficiencies highlighted above, the original design had many elements worth preserving in the improved UI. The most important one was the 'stateless operation' paradigm – all search results and cluster folders should be fetched in one request. With this approach the users would not experience any delays caused by additional communication with the server while browsing the folder list.

2.2 Prototyping of the new design

To allow ourselves a large degree of flexibility while brainstorming and polishing the new UI, we decided to use a prototyping technique based on static HTML files. This approach let us keep the cost of changing the user interface to a minimum and perform usability testing before writing any code. Over a few weeks, we worked on the prototype in an iterative fashion, increasing its fidelity and fixing problems identified by usability tests. In the last step, we ensured that the prototype displayed correctly on the major web browsers. Below we highlight the most important changes we introduced to the Carrot^2 user interface.

Search results sources as tabs. To address the most important problem with the previous UI, we decided that different data sources should be represented as 'tabs' (Fig. 1(c)), like in conventional search engines. The choice of a clustering algorithm, previously integrated with the search source, was delegated to a separate UI component. With this approach users can try all possible combinations of data sources and clustering algorithms.

Hidden clustering algorithm choice. As the majority of end-users will not have enough technical knowledge to purposefully select the clustering algorithm, this choice is only available in the block of 'advanced options', shown after selecting the *Show options* link placed next to the query field.

Polished startup and results pages. The redesigned user interface offers an enhanced startup page with a one-sentence description of what Carrot^2 will do with the query typed in by the user. The search results page, apart from the introduction of search source tabs, has not undergone any major paradigm changes. We preserved the previous layout (Fig. 1(d)) and stateless behaviour of the original UI. Minor improvements included: enabling independent scrolling of search results and folders by putting them in separate internal frames, making folder selection easier by slightly increasing the font size and spacing, adding icons to strengthen the 'foldering' metaphor. The the choice of search results sorting method was removed entirely, and the results are always shown in the order they were received from the search engine.

2.3 Usability testing of the new design

Following the experiences of Jacob Nielsen [3], who showed that testing a UI even with a handful of people can detect the most important design flaws, we performed a number of informal usability tests of the new UI. For each test participant, we briefly introduced the Carrot^2 clustering engine and explained

the goal of the tests, emphasizing that it is the UI that is being evaluated and not the participant's actions. Then, we asked to perform a number of search tasks using the UI, e.g. *query Google for 'data mining' and find results related to the field of text mining*. Throughout the tests, we did not help the participants to complete their tasks, but only carefully observed their actions and took notes of their interaction with the system.

The most important observation we made during the usability tests was that for some specialised data sources the users did not know what type of queries they could submit. To give such users some guidance, on the startup page we included example queries for each search tab.

2.4 Implementation and evaluation

An important input for the implementation phase was the static HTML prototype of the UI. With a high-fidelity prototype covering all the details of the UI, the implementation work could concentrate on providing an efficient technical backbone for the already designed graphics and interaction schemes.

A rather imperfect (but valuable) test of the impact of the new design is comparing the number of queries the public demonstration of Carrot^2 handled before and after the UI change. Before the first beta release of the improved user interface, which happened in August 2006, the monthly number of queries did not exceed 5 000. The query rate soared to over 13 000 in August 2006 and it further increased in September, when the new release of Carrot^2 was officially announced. Although the numbers at the end of 2006 may still to some extent be a result of the new release advertising, the popularity of Carrot^2 on-line demo seems to have stabilised at a much higher level. In addition to pure numbers, we also received quite a few favourable comments from users who liked the improved concepts and graphical finish of the new UI, which is a very important motivating factor for future work on the project.

The whole process of user interface redesign took a total of about 200 working hours spent between March and September of 2006. The two largest work items involved were building a prototype of the new UI and its actual implementation. The former took about 80 working hours spent between March and June of 2006, while the latter required about 120 working hours in the period between July and September of 2006. Compared to the total effort put into Carrot^2 since 2002, the UI-related work does not seem very costly.

3 The success factors of usability practices in Carrot^2

While open source developers declare that usability has a high or very high priority for them, they rarely apply it in practice with respect to their own projects [4]. One reason for this is, as explained in [5], that *most work on open source projects is voluntary, developers work on the topics that interest them and this may well not include features for novice users* and usability. Below we

summarise the factors that we found important for a successful introduction of usability practices to Carrot^2.

Usability enthusiast among project developers. One of Carrot^2's developers, in addition to being interested in web and text mining problems, is a usability enthusiast. This made the voluntary work on user interface aspects both interesting and rewarding. Moreover, by combining the role of a developer knowing Carrot^2's internals and a usability expert, we could avoid, for example, designing a UI that is impossible to implement due to technical reasons.

*Usability work was part of a major system architecture change.*The user interface redesign was carried out as part of a major refactoring of the system core. As a result of the change, we would have to make a large number of changes to the implementation of the original UI. This gave us a good reason to completely abandon the original UI and its implementation and create the new one from scratch. Unfortunately, throwing away large parts of code is a situation which many open source projects, especially those with public programming interfaces (APIs), simply cannot afford.

Very simple user interface. The new Carrot^2's UI we designed contained only two screens, which made it relatively easy to pay attention to every detail. For projects with larger UIs or a large number of distributed contributors, it may not be possible to carry out the usability work in a similar way.

4 Conclusions and future work

In this paper we described the process of introducing usability practices to a small OS project called Carrot^2. A number of favourable factors, such as having a project member who shared the developer and usability expert roles, contributed to the success of the UI improvement process. One lesson we learned during our work on usability is that creating a high-quality user interface does not necessarily have to cost much and can bring substantial benefits.

References

1. Relevantive: Linux Usability Study Report. On-line: http://www.linux-usability.de/download/linux_usability_report_en.pdf (2003)
2. Weiss, D.: Carrot^2: Design of a Flexible and Efficient Web Information Retrieval Framework. In: Proceedings of the Third International Atlantic Web Intelligence Conference, AWIC-2005, Łódź, Poland. Volume 3528 of Lecture Notes in Computer Science., Springer (2005) 439–444
3. Nielsen, J.: Why You Only Need to Test With 5 Users. On-line: http://www.useit.com/alertbox/20000319.html (2003)
4. Andreasen, M.S., Nielsen, H.V., Schroder, S.O., Stage, J.: Usability in Open Source Software Development: Opinions and Practice. Information Technology and Control **35A**(3) (2006) 303–312
5. Nichols, D.M., Twindale, M.B.: The Usability of Open Source Software. First Monday, On-line: http://www.firstmonday.org/issues/issue8_1/nichols/ **8**(1) (2003)