Using Concept Maps to Evaluate the Usability of APIs

Jens Gerken
Human-Computer Interaction Group,
University of Konstanz, Box D-73
78457 Konstanz, Germany
jens.gerken@uni-konstanz.de

Hans-Christian Jetter
Human-Computer Interaction Group,
University of Konstanz, Box D-73
78457 Konstanz, Germany
hans-christian.jetter@uni-konstanz.de

Harald Reiterer
Human-Computer Interaction Group,
University of Konstanz, Box D-73
78457 Konstanz, Germany
harald.reiterer@uni-konstanz.de

CHI 2010, April 10–15, 2010, Atlanta, Georgia, USA.
ACM 978-1-60558-930-5/10/04.

Abstract
Application programming interfaces (APIs) are the interfaces to existing code structures, such as widgets, frameworks, or toolkits. Therefore, they very much do have an impact on the quality of the resulting system. So ensuring that developers can make the most out of them is an important challenge. However standard usability evaluation methods as known from HCI have limitations in grasping the interaction between developer and API – the GUI, which makes this interaction obvious, is missing. In this paper we present a longitudinal approach using concept maps and a question diary to make this interaction visible and study the usability of an API over time.

Keywords
API usability, longitudinal evaluation, concept maps.

ACM Classification Keywords
H5.2. [Information Interfaces and Presentation]: User Interfaces.

General Terms
Experimentation, Human Factors.

Introduction
Developing a software system nowadays hardly means programming everything from scratch. Instead, developers can often rely on existing widgets,

frameworks, libraries, or software development toolkits that provide existing code structure for reuse. To access these, application programming interfaces are provided (APIs) and while there may be many different kinds of APIs they all serve the same purpose, as Daughtry et al. [4] described it: "they each provide a programmatic user-interface to a module of code". As with any kind of interface, some of them are more usable than others and so in recent years, the study of the usability of APIs has been emphasized by more and more researchers [e.g. 4,5]. We can identify two main goals in such studies. One is to analyze the usage of APIs on a more general level in order to derive design principles for the creation of new APIs or the modification of existing ones. The second is to evaluate the usability of one specific API, preferably during its development process as part of a user-centered iterative lifecycle. In this paper we will primarily focus on the latter part by presenting a longitudinal evaluation method that can be used to assess the barriers developers come across when trying to use an API as well as their evolution over time.

Challenges for the Evaluation of an API

Evaluating an API is quite different from standard usability evaluation. The most important aspect is the missing GUI, so using and interacting with an API is much more subtle than using a standard software application and therefore more difficult to observe and analyze. Accordingly, it is not straight forward to define wrong doings or errors during the observation of users since there are many ways to reach a goal. From a methodological point of view, the most common approaches are lab based usability tests in combination with thinking aloud. So for example Klemmer et al. [10] presented such a usability study with seven participants using the Papier-Mâché Toolkit for developing tangible user interfaces. Participants were first introduced to the toolkit and then were asked to complete three typical tasks by using it. Thinking aloud as well as participants' Java code was then used to analyze the usability of the toolkit. In a similar way, Heer et al. [9] analyzed the usability of their prefuse toolkit. An interesting alteration of this approach was proposed by Beaton et al. [2]. Participants first have to write in pseudo code what they would expect in the API for a certain task and then perform the real task using the API. Thereby, one can better assess the mapping between the user's mental model and its matching with the real world.

Quantitative measurements in such studies often are task-completion times [5,1], sometimes lines of codes [10], or number of iteration steps needed [1]. While these can help in comparing different APIs [5] they can only indicate usability issues in a rather broad sense. More detailed analysis of the think-aloud protocol and video observation data help in identifying more hidden usability issues. Nevertheless it is quite difficult to find themes or clusters of usability problems. One possibility is to use the approach taken by Clarke [3]. He used the cognitive dimensions framework and adapted it to fit the needs of API usability evaluation. By using this framework, evaluators can cluster findings in the different categories and by that get help in identifying which higher level concept of the API might be problematic. Ko et al. [11] on the other hand identified six learning barriers of an API in a large field study which can be again used to cluster qualitative data. Identifying such learning barriers can be one step to assess the threshold of an API, which basically means how difficult it is to achieve certain outcomes with it, as Myers defined it in their threshold and ceiling quality

Figure 1: Example for a concept map and the setting in general after the last session

criteria [12]. The ceiling defines what is achievable with an API. Common approaches to evaluate an API on this matter are case studies that show a wide range of possible systems [e.g. 9,10]. So while there is quite some work regarding the analysis of such API usability studies, we think that from a methodological point of view, there is still room for improvement. Current approaches seem to be insufficient to address two major aspects: 1) since most studies are limited to one or maybe a few hours of testing, tasks are rather simple and most of the time "pre-defined". More complex or even "free" tasks, where developers can use the API for own projects are seldom and difficult to integrate in such study designs, although such tasks would provide very valuable input regarding the usability of an API in real world situations. 2) It seems difficult to assess learning barriers or the threshold of an API during such a cross-sectional study. One would assume that barriers shift during longer usage times and thresholds may be perceived differently after some time. Both of these aspects can be addressed by using a longitudinal study design, which basically gathers data at more than one point in time [7], therefore making it possible to integrate more complex tasks and observe changes. In the following section we will present the concept map method to address these issues. It is based on a longitudinal field study design and a visual representation of the API usage.

The concept map method

The foundation of our method is the concept map approach: We observe pairs of developers during regular sessions, in which they have to draw a map which shows the relationship between the system they are building and the API (see figure 1). To structure this, participants are asked to label post-it notes and place these on a large sheet of paper (see Figure 1). This concept map thereby shows not only what parts of the API the developers are using but also how they think these parts work together. We rely on teams of two persons, as so they have to discuss the design of the map in group and talk aloud, giving valuable insights into the understanding of the API. The graphical representation of such a concept map makes it much easier to track changes over time. To do so our method asks participants to refine their map from session to session by sticking new post-it notes above existing ones or moving them around. So while it is difficult for users to precisely answer the question "how has your perception and use of the API changed during the last week", we argue that it is much easier to change the visual representation of exactly that question. The time frame for such a study obviously depends on the project scope and the ceiling of the API, but we suggest it to be at least several weeks to allow the investigation of changes over time.

In order to be able to get a grasp of the experience during the actual usage of the API we contemplate the concept map with a remote data-gathering technique. Based on the diary technique [13] and the question-suggestion protocol which was introduced by Grossman et al. [8] we use a questioning-diary. The diary allows us to be able to get data from the users in their normal work environment without them having the feeling of being observed. We suggest giving participants the feeling of being able to use the diary as a help line by stating their problems in terms of questions. The concept map sessions should then also include a suggestion/feedback part where questions are discussed and if possible answered by API experts/ designers. Thereby it is possible to analyze the API to a

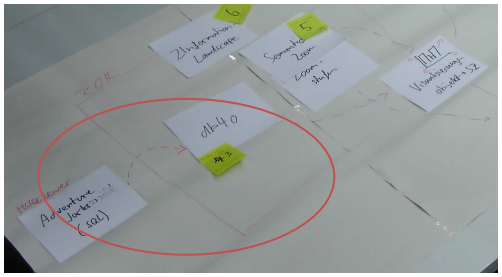

Figure 2: The initial concept map using the db4o backend for data storage

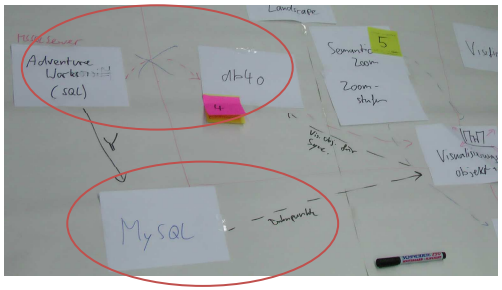

Figure 3: The evolution of the concept map, including an additional MySQL database. The changes are made explicit by crossing out earlier links between the data source and the db4o and introducing a link back from the visualization to the db4o.

much deeper degree since we avoid the danger of participants being stuck right at the beginning. Implementing the diary can be done in many ways while one of the easiest ones is to use a Wiki. This provides the possibility for participants to not only state questions but also update them over time if solutions come along, again making it possible to track and analyze changes. Both of these techniques allow for further modifications and fine tuning. Therefore, what we have presented so far should be seen as the abstract representation of the method. In the next section we will illustrate a concrete "implementation" of this method in a case study which furthermore points out the strengths and current weaknesses.

The ZOIL Case Study

Longitudinal studies in the field of HCI are still seldom applied, although the awareness and the need for such studies are constantly increasing. In order to establish a common methodological basis for longitudinal research in HCI Gerken & Reiterer [7] presented a taxonomy, which basically shows the design space for a longitudinal study. We will refer to this taxonomy for structuring throughout this case study.

The ZOIL API

The **Z**oomable **O**bject-Oriented **I**nformation **L**andscape (ZOIL) API, which was developed by one of the authors, provides access to the ZOIL framework, which is deployed as a software framework written in C#/XAML for .NET & Windows Presentation Foundation (WPF). It provides programmers with an extensible collection of classes covering a wide range of functionality, e.g. ZUIs, client-server persistency, and input device abstraction. Basically, it serves as a toolkit for developing zoomable user interfaces in the context of reality based interaction and surface computing [6].

Research Objective, Research & Data-gathering Design

Our primary research objective was the identification of walls and barriers when using the API and how these might change over time. In order to be able to have a rather broad range of API usage we opted for a multiple Case Study design, similar to the MILC approach [14]. We observed four groups of two persons each with different projects. The goal was in all cases to create a running high-fidelity prototype in the context of reality based interaction and surface computing, for which the ZOIL framework is meant for. So both, data and tasks were not pre-defined by the researchers but defined by the users themselves. This allowed us to observe the API usage under real conditions without the bias pre-defined tasks and data usually introduce into usability studies of APIs. Our test-developers were students of a lecture about visual information seeking systems. We used a five week period in which we conducted four regular concept map sessions on each Wednesday, separately for each group. The diary allowed further-more for event-based, on-the-spot waves of data-gathering throughout the time frame.

Data-gathering Methods

We will provide some more details on the implementation of our two data-gathering methods. First regarding the concept map approach, we asked our participants not only to place and link concepts but also to rate them on a rating scale from 1 "I don't like this concept at all" to 7 "I really like this concept". Besides, they had to express their ratings by using adjectives describing the concept, such as "neat" or "inconvenient". Again during each session we asked

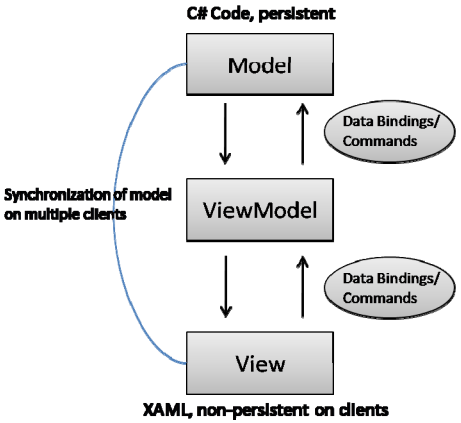

Figure 4: A challenge for the design of the ZOIL framework was to provide a client-server-architecture for a persistent and distributed ZUI, with each client providing the user with an individual view on the shared workspace. For this reason, the ZOIL framework uses the **Model-View-ViewModel Pattern (MVVM**) to provide a MVC-style separation between the persistent data model of an object and the non-persistent view of the object in the zoomable information landscape. Each object's model is shared via the server with all others clients, but the corresponding view is not shared and only resides on the client-side.

them to refine the ratings and the attributes they had given, making it possible to track changes over time. During the last session we gave our participants additional concepts of the API and asked them to place those within their map and to explain whether and how they made use of them. This allowed us to further elaborate their understanding of the API and the underlying framework.

Data Analysis

Both, the diary and the concept map technique were designed in a way to allow us to analyze changes in qualitative data over time. The focus is on making those changes visual and explicit, e.g. by asking the users to refine the ratings and adjectives and refine the structure of the concept map during each session. This lead to drastic and easily perceivable changes, such as concepts being shifted, removed, or renamed and questions in the diary being resolved and marked as such.

First Results & Outlook

In this section we illustrate the potential of the method and also some observed shortcomings of the current implementation.

The diary as a wiki still provided quite a barrier to use it on a regular basis. This resulted in small problems being not reported on the diary and sometimes the status of larger problems did not get updated over the course of the study, making it difficult to judge the reliability of the entered data. We suggest using a less obtrusive technique by integrating the diary directly within the integrated development environment (IDE) or the toolkit. An interesting approach could be the use of a twitter based technology as a very low-obtrusive and low level diary technique. Besides, our participants had problems in finding the right words to describe concepts of the API with adjectives. One possible solution to this could be to provide adjectives in advance, both negative and positive ones and then let participants simply choose from this sample.

The Concept Maps proved to be very useful to visualize the users' mental model of the API. In some cases, participants included deeper concepts of the underlying ZOIL framework within their maps, suggesting that they expected these still to be part of the API. Furthermore the discussions between the group members while creating the concept maps were extremely helpful in understanding their understanding and usage of the API. The analysis of the Concept Maps over time provided tremendous insights. One example is the database connection provided by the API, which actually is not used for data-storage but for the synchronization of the system on multiple screens. One group had problems in understanding this issue so in their first concept map, albeit having the need for data storage they only included the db4o backend connection provided by the API. In the second week they realized that this won't be sufficient and therefore included an additional concept for a MySQL database and used this in combination with the provided db4o backend (see figure 2 & 3). An example for the identification of a learning barrier is the usage of the Model-View-ViewModel (MVVM) pattern provided by the API (see figure 4). As it turned out, one group had difficulties for weeks to integrate this and in the very end came up with their own solution. However, the concept maps showed (figure 5 & 6) that they ended up with a very similar structure providing the basic separation between view and model. So while the

Figure 5: The original concept map implementing the work around the MVVM pattern

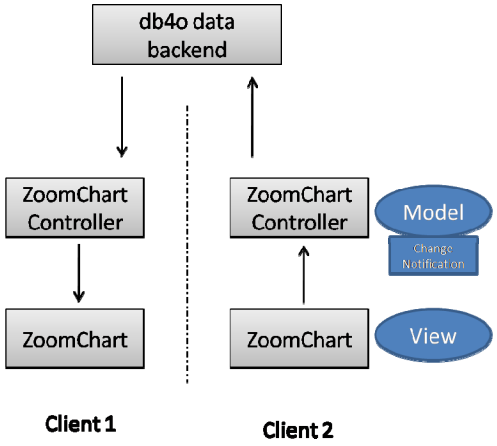

Figure 6: A schematic view of the implementation. They implemented a solution without a ViewModel, thereby losing one of the key advantages of the MVVM pattern – the abstraction layer between C# and XAML code.

MVVM pattern provides the necessary functionality, it can be clearly ascribed with being a learning barrier of the API. The question-diary still was useful for a discussion trigger during the concept map sessions, helping us to get more insight into specific problems. The additional task which we integrated in the last session that asked participants to place API concepts within their respective maps also gave additional insights into the thinking and level of comprehension of the API.

Outlook

We presented a longitudinal approach to evaluate the usability of an API. The method comprises a visual concept map technique and a question-diary as main data-gathering techniques with the focus on the ability to track changes in a longitudinal design. While our first results are promising, we will further refine and extend the method and apply it to a larger and more structured case study in order to be able to elaborate in more detail on the potential benefits. One potential design change will be a reduction of users' degree of freedom in creating the map to allow some additional quantification of the results which might be easier to grasp in some cases. We will also compare the approach to traditional API usability testing approaches.

References

[1] Ballagas, R., Memon, F., Reiners, R., and Borchers, J. iStuff mobile: rapidly prototyping new mobile phone interfaces for ubiquitous computing. In *Proc. CHI 2007*, ACM Press (2007), 1107-1116.

[2] Beaton, J. K., Myers, B. A., Stylos, J., Jeong, S., and Xie, Y. Usability evaluation for enterprise SOA APIs. In *Proc. of SDSOA '08*. ACM Press (2008), 29-34.

[3] Clarke, S. Measuring API Usability. In *Dr. Dobbs Journal*, May 2004, S6-S9.

[4] Daughtry, J.M., Farooq, U., Myers, B.A., Stylos, J. API Usability: Report on Special Interest Group at CHI. *Software Engineering Notes July 09*. ACM Press.

[5] Ellis, B., Stylos, J., and Myers, B.A. The Factory Pattern in API Design: A Usability Evaluation. In *Proc. ICSE 2007.* ACM Press. pp. 302-312.

[6] Gerken, J., Heilig, M., Jetter, H.-C., et al. Lessons learned from the design and evaluation of visual information-seeking systems. In *International Journal on Digital Libraries*. 2009.

[7] Gerken, J. and Reiterer, H. Eine Taxonomie für Längsschnittstudien in der MCI. In Proc. *Mensch & Computer 2009.* Oldenbourg Verlag. 2009. English summary/translation available here: http://hci.uni-konstanz.de/elmuse/longitudinal_taxonomy.pdf

[8] Grossman, T., Fitzmaurice, G., and Attar, R. A survey of software learnability: metrics, methodologies and guidelines. In *CHI '09*. ACM.

[9] Heer, J., Card, S. K., and Landay, J. A. prefuse: a toolkit for interactive information visualization, In *Proc. CHI 2005*. ACM Press (2005), 421-430.

[10] Klemmer, S. R., Li, J., Lin, J., and Landay, J. A. Papier-Mache: toolkit support for tangible input. In *Proc. CHI'04*. ACM, New York, NY, 399-406.

[11] Ko, A. J., Myers, B. A., & Aung, H. H. Six learning barriers in end-user programming systems. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on* (2004), pp. 199-206.

[12] Myers, B., Hudson, S. E., and Pausch, R. Past, present, and future of user interface software tools. *ACM Trans. Comput.-Hum. Interact. 7*, 1 (2000), 3-28.

[13] Rieman, J. The diary study: A workplace-oriented research tool to guide laboratory efforts. In Proc. *INTERCHI 1993*, ACM.

[14] Shneiderman, B. and Plaisant, C. Strategies for evaluating information visualization tools: multi-dimensional in-depth long-term case studies. *AVI 2006 Workshop BELIV '06*. ACM. 2006