Web Information Systems

Assignment 3:

Design and engineering challenges of *Web 2.0*

Radim Čebiš
October 2007

1. Introduction

Even though the concept of *Web 2.0* [1] is more than three years old, it is still much discussed topic. For some people *Web 2.0* represents only another marketing buzzword, for others it means the real revolution of the world wide web. I am not here for judging their opinions but rather to point out what features are behind *Web 2.0*, how we could use them in the area of the web information systems and how they are used now in most popular applications.

Web 2.0 is just a term used to describe new trends in development of internet applications. Some of the trends might be tightly coupled with the progression of the internet. We should not ignore that in year 1997 only 70 million people were connected but nowadays more than 1.2 billion people are connected to the internet [2]. This is important for some *Web 2.0* aspects which I will cover in the next section. Also we should not miss out that connection speed has increased more than 10 times. Without this progress there would not be such interest in many popular services which are now considered as *Web 2.0*.

There is a lot of new approaches in *Web 2.0* which are very unusual for common software development. But I see the most challenging part of *Web 2.0* in new business models. *Web 2.0* is counting more on the users which are often considered as co-developers. This disallows the companies to charge them. Pricing of the services and investments into *Web 2.0* applications are very interesting problems of this new approach, but these are not our concern.

2. *Web 2.0*

It is harder to describe what the *Web 2.0* means because it has not strong boundary. It also includes more trends which makes it impossible to describe in one sentence. At least we can say that one of the key principle is that the web is used as a platform. Applications are delivered entirely through a browser. This leads us to another aspect of *Web 2.0* - rich internet applications. Interface had to be improved for better usability and interactivity, to be able to compete with standard desktop applications. Interface is usually based on *AJAX* [3] (Asynchronous *JavaScript* and *XML*) which is group of several technologies allowing application to send, receive and process data from the server in the interactive style (without a need to reload the page).

Because the application is run on the owners hardware, it is possible to release new versions as often as needed. The end of the software adoption is often perpetual beta – software never leaves the development stage of beta which allows improvement of the service even on the

daily basis.

But in my opinion the most interesting idea behind *Web 2.0* is collaborative content. Users are encouraged and given means to contribute their knowledge or data. This made a border between reader and writer blurred. By the means I mean user-friendly systems for publishing and sharing, but also supporting systems to encourage users, for example ratings and reputation systems. The collaboration would not be successful, if there had not been so many people connected to the internet. Of course this dependence on users brings new problems. One of them is cold start – we need users to create value of the web, but it is hard to start when there is no content. Also every day huge amount of content appears, but most of it is just worthless. The difficulty of navigation in increasing amount of data is usually handled by tagging. Also quality requirements are met with rating.

3. *Web 2.0* features in popular applications

I will start pointing out *Web 2.0* features in nowadays web applications and their impacts.

My first examples are concerning rich user interface. The first one is *Google Suggest* [4]. It is simple demonstration how *AJAX* can be used for empowering interface with understandable interactivity. It did not take much time and similar functionality was implemented on other webs which offer the user a searching.

Other very useful application of *AJAX* is employed by *YouTube* [5]. In this case we are offered almost all functionality we need just by using *AJAX*. For example we can list the discussion, save video to favourites or rate it and so on. The need of *AJAX* at the *YouTube* pages with videos is obvious. The video is not interrupted by our activity because we do not need to reload the page. I would always think about this approach while designing web application using video or audio playback.

There is a lot of examples of rich user interface. Even though *Gmail* [6] might be one of the most known. I will rather focus on *Google Docs* [7]. This application includes web-based word processor and spreadsheet. It supports versions and sharing, allowing users to collaborate on editing files. I downloaded the *JavaScript* source code for spreadsheet editor. As usually it is obfuscated, but when we reformat it so the length of one line is less than 80 characters, we end up with a text file more than 6700 lines long. If we take into account that names of the variables are after obfuscation just a few characters long and that programmers usually end lines just after one statement, we can guess that the source code in original readable version has more than two times more lines. This shows how complexity of web applications can be shifted to the client-side. Of course not every application needs spreadsheets, but it shows that much more effort is spent on the user interface.

Another interesting feature of *Web 2.0* is collaborative content. In my opinion the most known example is *Wikipedia* [8]. It is an encyclopedia which is written by thousands of volunteers. Articles can be edited by anyone and it supports reversal of editorial mistakes. Of course many of new problems can arise with this kind of freedom, but *Wikipedia* has its ways how to handle disputes and abuse. Intentional vandalism can be reported and corrected by anyone. Also new users' votes are given less weight in some polls, until they establish themselves in the community.

Next popular site using collaborative content is *YouTube* with users sharing their videos. *YouTube* also supports a lot of functionality for social networking and encourages their users to communicate. User can have “friends” and other people can subscribe for his videos – meaning

they will get notification that someone has published a new video. There are also groups allowing multiple people to discuss things publicly and post videos that apply to discussion. User can also post video response for another video, rate or discuss any video.

The use of collective intelligence can be also find in the way *Google* searches the Internet, because its algorithm for ranking a page rely on how many links the robot can find to the page, meaning it does not only use content of the page but also tries to guess how much people like it according to how often the page is mentioned elsewhere. This can be considered as using intelligence of all content creators of the web.

The impact of the collaborative content is obvious – there is much more of the data on the web. That is why tagging became necessary. Without tagging the users would be lost in chaotic labyrinth of data. That is why users tag their photos on *Flickr* [9] or their videos on *YouTube*. Users use their own language which is most probably understandable for other ones. Usually users can discover who created a given tag. Many tags are often used to describe data. Then the tags are used for navigation. For example we can find tag clouds on *Flickr* where the most often tags are written with bigger fonts.

4. Application problems and challenges

Rich user interfaces bring a new challenges for web developers. Now they need to write sophisticated scripts as opposed to easy controls of the forms used in the past. I was implementing *Google Suggest* feature for two companies in the Czech Republic. From my experience I can say that next time I would use some *JavaScript* library, because working purely with *xmlHttpRequest* and processing *XML* must be sometimes handled differently in various browsers. Fortunately we can find a lot of libraries trying to handle most incompatible problems. In my opinion *JavaScript* was not designed for such a huge programs where we have to use libraries. It is prototype-based language meaning that we can override any function's behaviour. This is used for example in *Prototype* [10] library which makes it hard to integrate with other libraries which are using built-in functions and rely on the standard functionality which can be corrupted by the *Prototype* library. Also programmers must think more about asynchronous access to keep a consistency of the page. Security problems should be also considered.

I think there are not big problems with implementation of collaborative functions. It is more the design challenge. We should think hard about how to prevent abuse and how to support navigation in huge amount of heterogeneous data. In discussions we can try to implement some kind of filter of vulgarities and so on. Navigation can be supported by tagging which is not hard to implement. I would say that mostly implementation should not be a problem and we always can steer our prevention of abuse on the run. Sometimes it is possible that there are problems with intellectual property, but I see bigger problem in the cold start – we have to attract other people to our page and then convince them that it is useful to contribute. This can be hard in the start when we do not have much of the content. And it becomes even harder when we want to start application focused on “small” group of people for example for people not speaking international languages. But I feel that this is more of the challenge for marketing people.

5. Conclusion

Web 2.0 tries to draw attention to some ideas. I think we can use a lot of them but we should not be always pressing on them. It is a nonsense to apply collaborative approach when we

want to create web presentation of the company, but it can be useful to create rich user interface. In the intranet information system it might be useful to use some aspects of the collaborative approach, for example tagging. Of course we can see that a lot of features have been used before *Web 2.0*. For example in intranet information systems we usually find information about who published what.

Even perpetual beta is not such a new think. We can read old papers about web engineering pointing out the constant change. We also have agile methods for software development supporting perpetual beta which were invented before *Web 2.0*.

I think *Web 2.0* encapsulates some ideas about which we should think, but it is always about concrete application to debate about them. Sometimes it can be useful to implement some kind of compromise for example support of fixed categories and tagging. I think it is always useful to think about ideas behind it and how it can be useful in our concrete application than to force it just because it became popular marketing word.

References

[1] What is Web 2.0 by Tim O'Reilly
(http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html)

[2] Internet Usage Statistics
(http://www.internetworldstats.com/stats.htm)

[3] AJAX (programming)
(http://en.wikipedia.org/wiki/Ajax_%28programming%29)

[4] Google Suggest
(http://www.google.com/webhp?complete=1&hl=en)

[5] YouTube
(http://youtube.com)

[6] Gmail
(http://gmail.com)

[7] Google Docs
(http://docs.google.com/)

[8] Wikipedia
(http://en.wikipedia.org/)

[9] Flickr
(http://flickr.com/)

[10] Prototype JavaScript framework
(http://www.prototypejs.org/)