The Nokia Open Source Browser

Guido Grassel[1], Roland Geisler[2], Elina Vartiainen[1], Deepika Chauhan[2], Andrei Popescu[1]

[1]Nokia Research Center, P.O. Box 407, 00045 Nokia Group, Finland
[2]Nokia Technology Platforms, 5 Wayside Road, Burlington, MA 01803, U.S.A,
[guido.grassel, roland.geisler, elina.vartiainen, deepika.chauhan, andrei.popescu]@nokia.com

ABSTRACT

With the advent of faster wireless networks and more capable mobile devices we expect to see growth in the mobile use of the Internet. In this paper we describe a new Web browser for mobile devices that we have built based on Open Source Software components. Our goal was to design a full Web browser that is easy to use, an architecture that is portable to other mobile software platforms, and an Open Source development approach to give others the opportunity to further develop it or use it for research purposes. We describe our technical implementation, the usability features that we invented, and discuss the benefits and Nokia's plans to work with the Open Source community to further develop the browser.

Keywords

Web browser, Internet, Mobile devices, Usability, User Experience, Open Source Software.

1. INTRODUCTION

We started looking into a new Web browser for the S60 smart phone mobile software platform[1] of Nokia in 2003. It was not enough that it would show pages authored in a special, mobile friendly, syntax such as XHTML Mobile Profile [16], and a mobile-optimized layout. We set out to solve several challenging goals at the same time:

First, we were aiming for a Web browser that would be capable of showing all Internet Web pages. Effectively, we were aiming for the level of Web site compatibility of mainstream desktop browsers.

Second, we wanted to solve the usability problem of browsing Web pages on mobile devices that had been created with consideration of desktop computers only. This meant both solving the issue of showing a large page on the small screen of a mobile phone, as well as finding highly usable means how the user could navigate on the page and interact with it. We were not satisfied with the state-of-the-art and were looking for a novel Web viewing method. A solution to the first problem was required for solving the second problem because excellent Web site compatibility is a pre-requisite for good usability.

Third, we wanted to bring down costs for our own software development and for licensing third party software.

This paper is structured as follows: Chapter 2 focuses on the background of our research by outlining related work. Chapters 3 and 4 describe our browser implementation work and the usability features we developed. Chapter 5 discusses our plans to work with the Open Source community, chapter 6 outlines our lessons learned and benefits, and chapter 7 summarizes and makes final conclusions.

[1] http://s60.com

2. RELATED WORK

Both Web browsers licensed by Nokia as well as S60's own browser used *Narrow Layout*. Narrow Layout is a method whereby the Web page is reformatted into one column that fits the width of a typically small handheld device display. This way, the need for horizontal scrolling is eliminated and the user will see all the content just by scrolling down. From our own experience using these browsers, and based on usability studies [17] we concluded that this method was insufficient. The main concerns with Narrow Layout were as follows:

- This method often destroys the intended logical grouping of content, leading to situations where users cannot even recognize familiar pages.
- It is hard for users to realize that they have proceeded to a new page after following a link, because the first screen of the new page may look exactly the same as that of the previous page.
- Pages that rely on a two-dimensional layout (e.g. timetables, maps) are broken since Narrow Layout will force them into a one-dimensional layout.

Based on these observations, we concluded that we had to find a Web viewing method that works better for end-users than Narrow Layout.

In addition to Narrow Layout, there exist several Web viewing methods for handheld devices that apply the O*verview plus Detail* method. In this method, an overview is used to display the whole data, while a detailed view shows a close-up portion of the data. These views can be presented next to each other [23], by showing them separately [14],[15], or by overlapping them [9]. If the views are shown simultaneously, with the overview on top of the detailed view, transparency can be used to avoid distracting the detailed view [13].

Implementations of the Overview plus Detail method can be divided into two groups: the ones that only visualize a Web page in a different way but do not modify its contents and the ones that make modifications to the page contents in order to optimize for small screen devices. The implementations that do not modify the page are based on showing the whole or a part of the overview of the page [2],[9],[23]. The page can be analyzed to create logical sections that can be selected by the user for viewing them independently from the rest of the document. Another solution is to allow the user to interact directly with the overview by means of special tools, such as a link selector, or a pick-up tool for extracting certain elements.

The implementations that modify the page aim to optimize and extract sensible content to be shown to the user [5], [8],[15].

An intermediate content transcoding proxy can be used to create the optimized content including summary views or to convert HTML pages [4],[12].

Apart from Narrow Layout and Overview plus Detail, other Web viewing methods simply eliminate some of the content [6], sometimes even without offering any possibility to view the page in its original form (layout and content) [24].

Some implementations are targeted for PDAs that have larger screen sizes than mobile phones [2],[4],[23],[25], while others require a pointing device [2],[9],[12],[15].

3. IMPLEMENTATION

The overall architecture of the new Nokia Open Source Browser is depicted in Figure 1. The core of the system consists of two Open Source cross-platform libraries, WebCore and JavaScriptCore, a library with platform dependent functionality, S60 WebKit, and the user interface. The role of each of these components is detailed in the following sections.

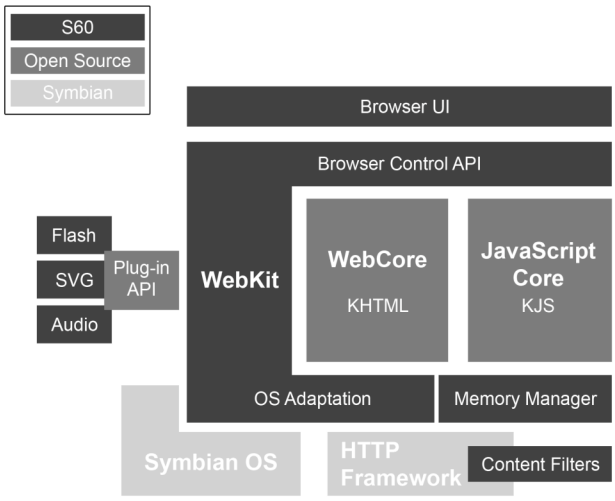

Figure 1: Nokia Open Source Browser architecture. Light gray boxes indicate components that are part of Symbian OS. Medium gray boxes designate cross-platform components, which are shared with Mac OS X, while dark gray boxes show S60 specific components.

3.1 Browser Engine Choice

We needed a very good browser engine for solving the Web site compatibility issue. It was not an option for us to develop our own browser engine if it was meant to be on a par with Mozilla Gecko and Internet Explorer Trident in terms of Web site compatibility. It requires many years of developing, testing and optimizing a browser engine in order to reach a comparable level of Web site compatibility. The main causes for this situation lie in the large variety of Web standards interpretations, as well as in the multitude of proprietary markup extensions.

Since we wanted to further develop the browser engine, we needed full access to the engine source code. That prevented us from working with a licensed engine since none of Nokia's licensing partners were able to provide the source code of their browser engine. We therefore decided to go with an Open Source browser engine and chose KHTML of the KDE Konqueror browser. The benefits of KHTML consisted of a clean design, small code size, good start-up performance, low memory consumption and sufficient Web site compatibility. We found our decision reinforced when Apple announced in January 2003 that they had made the same choice for their Safari browser [1]. We decided to start with Apple's code base, WebCore and JavaScriptCore, because we benefited from Apple's changes to the original KDE code to make WebCore and JavaScriptCore more portable, and from their improvements to performance and Web site compatibility.

3.2 WebCore and JavaScriptCore

Both components form the heart of the browser engine. WebCore is an Open Source, cross-platform library that contains the functionality required to parse, format and render HTML [21] documents. Its most important parts are the HTML and CSS [19] parsers, the HTML and XML [22] DOM (Document Object Model) [20] implementation, the document layout and rendering logic as well as main memory cache and a resource loading component. The main strengths of WebCore are clean design, portability, small code size, low memory consumption and, as a result of that, a short start-up time. Since this component has been extensively tested and improved over time, it now provides excellent compatibility with Web content.

JavaScriptCore is an Open Source, cross-platform JavaScript engine. JavaScriptCore is integrated to the DOM implementation in WebCore to access the HTML document and pass events.

WebCore and JavaScriptCore are both licensed under LGPL [7]. The main requirement imposed to a software component by the LGPL license is that any changes to the component itself must be released as Open Source. Separate software built around an LGPL component (e.g. the S60 browser application or S60 WebKit) is allowed to remain closed source.

WebCore and JavaScriptCore are used by Apple Safari, Apple Dashboard, and other applications in Mac OS. Apple engineers took the KHTML engine from the Linux KDE project as a basis for creating WebCore. They enhanced KHTML significantly and released the new version as Open Source Software. Unfortunately this resulted in a fork with the KHTML code base. JavaScriptCore is based on KDE KJS library and a fork of the code was avoided in this case.

3.3 S60 WebKit

The S60 WebKit library contains platform dependent functionality such as image decoding, graphics drawing, resource loading over Symbian OS's HTTP stack or from the device local storage, SSL and certificate management, etc. The WebKit library also implements a "browser control interface" that allows S60 GUI applications to include the browser as a normal control. WebCore and WebKit interact following a bridge-like pattern, where certain components defined in WebCore delegate operations to WebKit objects, which are part of a different class hierarchy. All code is written in C++.

4. USABILITY IMPROVEMENTS

Using an Open Source browser engine was not enough to achieve all of our goals. It solved pretty well the compatibility problem with Web content, but there were more usability problems for mobile browsing, such as how the user views, navigates, and interacts with the content. As discussed under related work, we felt that none of the state-of-the-art Web viewing

methods could solve the problem well enough or required a larger screen size or a pointing device. Therefore, we had to create something new and better. We followed a user-centric approach for which several user studies were the starting point. We iteratively created and refined a number of mobile optimized features, prototyped them, and evaluated their usability. Usability evaluations were first done by usability experts, and later testing the features with end-users and in the lab, and finally in larger field trials.

In the following we outline the features that achieved the best usability in our tests. These features are included in the new Nokia Open Source Browser that ships as part of the S60 3rd edition platform.

4.1 Unique Usability Features

Intended Web Page Layout

The Nokia Open Source Browser renders the Web page as intended by the author meaning that it obeys the CSS (Cascading Style Sheets) layout definition. The achieved layout is very similar to the one on the PC. A user who is familiar with a Web page from the PC easily recognizes the page when viewing it on the Nokia Open Source Browser. By rending a Web page with the intended layout we avoid the problems caused by reformatting the page in Narrow Layout: "breaking" pages that rely on a two-dimensional layout, with the consequence that users lose orientation, and need to scroll a lot vertically.

Narrow Text Column

Applying only Intended Web Page Layout can result in a paragraph of text being laid out wider than the screen. The user would need to scroll horizontally for reading each line of text in such a layout. To avoid this situation we have modified the CSS layout algorithm of the browser in such a way that the width of a line of text is at most as large as the viewport width (i.e. it is also never wider than the screen).

Figure 2a: In normal layout text runs wider than the viewport (marked by red rectangle).

This layout mechanism is a unique feature, we do not know of any other mobile browser with this or a similar feature. Figure 2a shows how text is rendered without the Narrow Text Column feature, and Figure 2b shows the same text with the Narrow Text Column feature enabled.

Figure 2b: The Narrow Text Column ensures the text width fits the viewport (marked by red rectangle).

Mini Map

The purpose of this feature is to provide the user with a sense of orientation and overview of the Web page. A thumbnail, or Mini Map, overview of a larger part of the page is displayed automatically whenever the user continuously scrolls the page in horizontal or vertical direction. The Mini Map is shown on top of the main view. A red rectangle in the thumbnail indicates the current visible part of the Web page in the main view. The thumbnail overview appears with a short delay while scrolling, and disappears immediately when the user stops scrolling. Figure 3 shows this feature in action.

Figure 3: Mini Map shown on top of the main view of a Web page.

Page Overview

The Page Overview serves the same purpose as the Mini Map: it provides a full-screen overview of the Web page. While the Mini Map becomes visible automatically, Page Overview needs to be activated and de-activated by pressing a shortcut key (e.g. "5" on a Nokia E60 keypad). By moving a red rectangle over the full-screen overview with the cursor keys, the user can re-position the browser viewport over the desired Web page region.

From our experience, Page Overview and Mini Map are complementary features. It is easier for users to discover the Mini Map (since it appears automatically), whereas Page Overview, due to its larger full-screen size, provides more context. Some users have commented that Page Overview works better on Web pages whose layout they are familiar with, while the Mini Map is

more suited for finding content on unfamiliar pages. A screenshot with this feature can be seen in Figure 4.

Figure 4: Page Overview.

Text Search

Because of the inherently smaller browser viewport, searching on a Web page is more important on a small screen device than it is on a desktop browser. We provide an easy way for the users to quickly search for a text string on the current Web page. While the user is typing, the viewport automatically scrolls to the first occurrence of the entered search phrase (Figure 5). Pressing the left soft key will offer a menu option that allows the user to move the viewport to the next occurrences (if any).

Figure 5: Text Search on Web page

Virtual Pointer

The Nokia Open Source Browser employs a sophisticated virtual pointer that the user controls with the 5-way joystick. Here, the innovation lies in combining the two state-of-the-art methods: focus navigation and mouse pointer. This way, we get the benefits of both methods: being able to easily and quickly select a focusable element with a minimal amount of key presses (with focus navigation), and the ability to freely position the pointer on top of any element (e.g. to enable a context sensitive browser menu). Our method therefore naturally supports dynamic content such as DHTML and CSS menus. Firstly, the optimization of a free moving pointer method lies in the intelligent adaptation of the distance the pointer moves each time the user presses the cursor keys. This distance changes depending on the distance to the nearest page element in the direction of the moving pointer. Secondly, the Web page scrolls automatically as the pointer moves over the page. By scrolling the page in vertical or horizontal direction the pointer remains close to the center of the viewport unless an edge of the Web page has been reached.

Visual History

Visual History is our improved version of the "Back" and "Forward" functions of browsers. The Visual History shows thumbnails of previously visited Web pages. It is activated by pressing the right soft key ("Back"). It allows the user to go one or several pages back (or forward) in the page history at once. Its main benefit is that it allows the user to quickly switch to a new page without waiting for intermediate pages in the page history to render. The Visual History and the Mini Map have been the two features that created the biggest "wow" effect in user tests and when demonstrating the browser. Figure 6 shows an example screenshot of the Visual History.

Figure 6: Visual History

Multiple Document Support

A required minimum level of multiple document support needs to handle popup windows. While we agree that Web pages should not use popups and popup blockers are useful, there are sites that do not work without support for popups. Full fledged multiple document support consists of several additional features: the possibility to open a page in a new window, to select a link to be opened in a new window, and easy means for switching between open documents. We have user-tested several solutions. Two solutions scored about equally well in our tests: First, tabs (Figure 7a) similar to Mozilla Firefox and Opera for PCs, and second, a combination of Visual History and document switching in one function (Figure 7b). More advanced users preferred the latter solution. One reason for their preference might be that document switching is readily available from the right soft key in this solution. The simplicity of the user interface was likely the reason why average users preferred the first solution. Initial releases of the Nokia Open Source Browser will only have basic support for popups and full multiple document support using either of the discussed solutions is expected later.

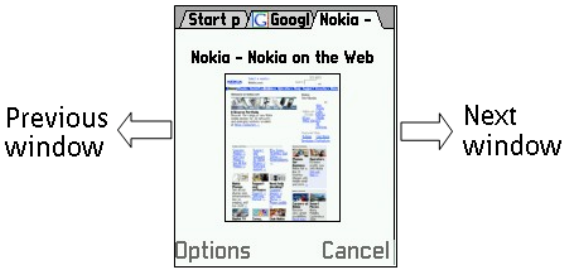

Figure 7a: Document switching with tabs

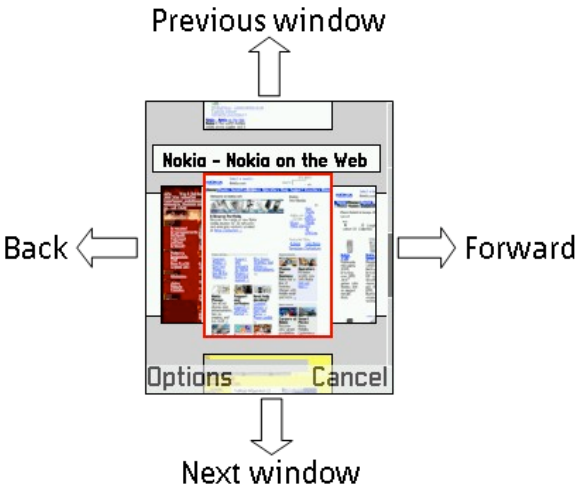

Figure 7b: Combined Visual History and document switching

4.2 Usability Test Results

As mentioned above, we have conducted a large number of end-user tests to refine and verify aforementioned features.

Intended Page Layout & Mini Map versus Narrow Layout

We carried out a field trial to verify the superiority of our own Web viewing method over Narrow Layout. The tested version of our own method included an implementation of Intended Page Layout, Narrow Text Column, Mini Map, and Text Search features on the research prototype of the Nokia Open Source Browser running on a Nokia 6600 GPRS phone. For testing Narrow Layout we used a commercial version of ACCESS Netfront running on the same phone to obtain comparable results.

This trial involved 20 users of various ages and backgrounds. We had 12 male subjects and 8 were female, ages 15-50. Of all users, 7 had never used a full Web browser on a mobile phone before, while 5 participants were frequent users. The trial lasted two weeks. Half of the test subjects used the Narrow Layout method first. The other half used our own Web viewing method first. After one week we swapped the Web viewing method.

After using the methods for one week each we asked which method they preferred for viewing Web content on a mobile phone. We used a 7-point scale 3 meaning strong preference for either method, and 0 meaning no preference. At the end of the trial, 18 participants preferred our own method, while 2 users liked the Narrow Layout method better. Usually in this kind of tests, it is rare to get participants give strong preference ratings, so it is notable that as many as 12 participants used the extreme preference rating for our own method. The order of testing the method affected the ratings, so that the first used method got their preference more easily: All users who used our own method first (Group 1) clearly preferred it, whereas the preference distribution of the other group (Group 2) varied more (Figure 8). Still, 8 out of 10 participants who first learned the Narrow Layout method preferred our own method after they also learned to use this one. Roto et.al. have published the full details of this test [18]. This result shows a clear superiority of our own method over the state-of-the-art.

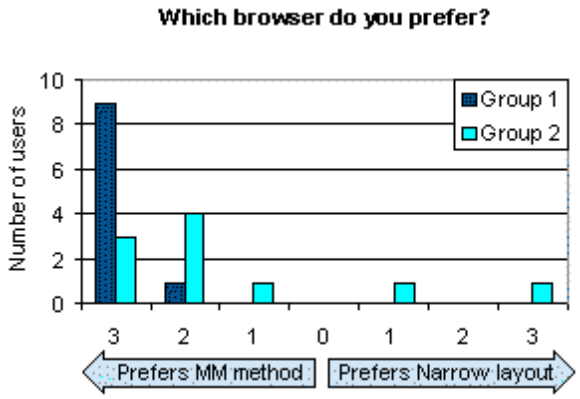

Figure 8: 18 participants preferred the Browser with the Mini Map method, Group 1 more clearly than Group 2.

Intended Layout & Page Overview versus Narrow Layout

The Page Overview feature was created later than the other features. Therefore, we executed another usability evaluation for this feature. This time, our implementation included the Intended Layout, Narrow Text Column, and Page Overview features. Implementation was again on the research prototype of the Nokia Open Source Browser on a Nokia 6600 phone. The method to compare with was again Narrow Layout. Users either used ACCESS Netfront or Opera Mobile if they had used Opera Mobile before. Users were allowed to switch between Narrow Layout and Intended Layout on Netfront or Opera Mobile.

We used 20 test subjects, 6 male, and 14 female, ages 21-50. These were other persons than in the previous test. The test was done in the lab, for at most 2 hours. All but one user had previous experience with browsing on mobile devices.

When asked if they needed to access Web pages via a mobile phone, which browser they would prefer to use 18 out of 20 persons preferred our own method. The preference of 10 was extreme (score 3). Only two users had a slight preference for the Narrow Layout method (Figure 9). Also these results show a clear superiority of our own method over the state-of-the-art Narrow Layout method.

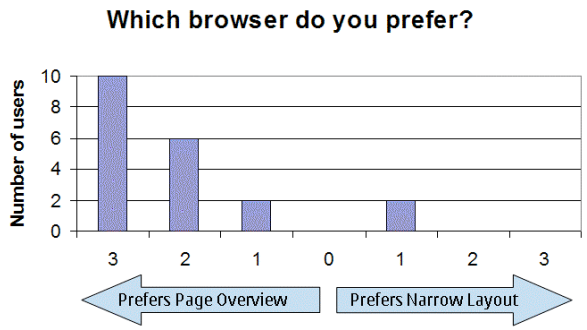

Figure 9: 18 participants preferred the Browser with the Page Overview and Intended Layout features, 2 users had a slight preference for the browser with Narrow Layout.

It should be noted that the results of the first and the second tests cannot be interpreted that Page Overview would be better than Mini Map because the tests differed in several aspects, e.g. different test setup, different test subjects, and different tasks. We have reason to believe the two features complement each other nicely and the product version of the Nokia Open Source Browser that includes both features scores even better when compared against the Narrow Layout method.

5. NOKIA OPEN SOURCE PLANS

Nokia is committed to Open Source, intending to actively participate in the Open Source community to further develop and enhance the browser, contributing Nokia's expertise in mobility. The Nokia Open Source Browser will ship on S60 3rd edition devices, some of which will be available in the first half of 2006. This browser is already available as part of the S60 3rd Edition SDK, available for download from Forum Nokia[2].

As the first step in our Open Source plans, we released our modified sources for WebCore and JavaScriptCore in order to comply with their LGPL term, which requires us to disclose any changes that we made to the licensed code. The modified source code for these components is available at the Nokia Open Source Browser Web site[3].

Next, we intend to host the sources for S60 3rd Edition WebCore and JavaScriptCore at the WebKit Open Source Project Web site[4]. This will enable the community to look at the sources in a source control system (preferred to zip releases) and to compare it with the sources in the tip of the tree. This will also allow Nokia developers to explain to the community some of the design decisions needed to port these components to mobile devices. We want to avoid a permanent branch WebCore and JavaScriptCore. Therefore, our goal is to merge our changes with the main development branch of the WebKit Open Source project.

As a final step towards contributing a complete mobile browser to Open Source, we will release our S60 WebKit source code and tools. The community then will have access to a complete mobile browser for S60: WebCore, JavaScriptCore, plus the supporting infrastructure, which includes S60 WebKit, Memory Manager, and a sample UI. Together these components enable developers to build, use, and test the browser on S60 phones, and on the emulator in the S60 SDK, running on Windows. The Nokia browser user interface, including some of the aforementioned user experience enhancements (such as Page Overview) will remain closed source. The Nokia Open Source Browser team intends to change its operational model so that Nokia developers can work on the browser code along with the Open Source community.

The combined development effort of Nokia, Apple, and the Open Source community on the Open Source browser engine will hopefully result in fast adoption of the new Web technologies as well as enhanced compliance with Web standards. This will result in a high quality, low cost browser engine that is available to all. This will encourage use of the WebKit browser engine by other mobile vendors thus reducing the browser fragmentation on mobile devices. Our hope is that the WebKit browser engine becomes the de facto Open Source engine for mobile browsers thus driving innovation in content, services, browser features and technology.

6. LESSONS LEARNED

As the first Open Source application for mainstream mobile devices in Nokia we learned valuable lessons from this project:

First, Open Source can offer high performing and standard compliant software also for embedded software solutions: WebCore provides fast rendering and scrolling support, and our new Nokia Open Source Browser is more than 117% faster than our own Nokia in-house browser. In addition, the excellent Web site support of WebCore and JavaScriptCore that had been developed over years in the KDE community and later by Apple resulted in the new Nokia Open Source Browser being 45% more Web compliant than our existing in-house browser.

Second, using Open Source reduces time to market and focuses company resources to innovate: It took less than two years for us from the time we decided in Nokia to build a new browser based on Open Source to the time we shipped our first products. The Web browser is also on mobile software platforms one of the most complex software components, and it would have taken many years if such an application would have been built in-house. In the past, many of our resources were tight to improve basic Web site compliance and integration of 3rd party browsers into Nokia products. By using Open Source code for the browser engines we could instead focus our attention to develop new innovative features, as presented in chapter 4, which improved end-user usability.

Third, if Open Source code is architected well then legal risks can be managed: Legal risks are usually one of the main reasons why companies hesitate to use Open Source Software in their products. As WebCore and JavaScriptCore are released under the LGPL license, diligent code review or “scrubbing” was needed before we could make our changes publicly available, to avoid that we would release any proprietary code or code that is subject to more limited licensing terms than LGPL. Also choosing a license for components that were developed in-house and need to be Open-Sourced requires diligent legal advice.

Finally, working with Open Source also for an embedded software application is rewarding and stimulates the innovative mindset: The prospect that some of the developed software will be

[2] http://forum.nokia.com

[3] http://opensource.nokia.com/projects/S60browser

[4] http://webkit.org

available as Open Source was seen as something very rewarding for many of our software engineers. The Open Source mindset is highly innovative as new ideas can openly be discussed and collaboration is not limited to your own company and partners.

The key benefits that we have seen from this Open Source mobile browser project over our in-house embedded software development and 3rd party licensing are in short: lower R&D costs, better resource focus on innovation, improved software quality and compatibility, and reduced time to market.

7. CONCLUSIONS

We developed a Web browser for mobile devices for the S60 software platform that is based on Open Source components. Our architecture is based on the Open Source browser engines WebCore and JavaScriptCore from Apple that were originally developed by the KDE community. To improve usability we developed a number of novel features that allow users to read Web pages on their mobile devices in a similar way like on their desktop. We presented Mini Map, Page Overview, Visual History and others, that provide superior usability to access full Web pages on mobile devices than existing methods.

We have thereby progressed beyond the state-of-the-art for mobile Web browsers in terms of usability as well as technical realization. By using Open Source engines we invite the Open Source and research community to work together with Nokia and others to improve browsing for future mobile devices that will be used by millions of people – in the future possibly more than desktop and laptop computers combined.

8. ACKNOWLEDGMENTS

The authors wish to acknowledge the contribution of their colleagues at Nokia Research Center and Nokia Technology Platforms to the work described in this paper. We want to specially mention Virpi Roto and Antti Koivisto.

9. REFERENCES

[1] Apple Computers press release on Safari launch: http://www.apple.com/pr/library/2003/jan/07safari.html

[2] Baudisch P., Xie X., Wang C., Ma W. *Collapse-to-Zoom: Viewing Web Pages on Small Screen Devices by Interactively Removing Irrelevant Content*. Proc. ACM UIST 2004.

[3] de Bruijn O., Spence R., Chong M. Y.: *RSVP Browser: Web Browsing on Small Screen Devices*. Personal and Ubiquitous Computing (2002) 6:245–252.

[4] Buyukkokten O., Garcia-Molina H., Paepcke A., Winograd T. *Power Browser: Efficient Web Browsing for PDAs*. Proc. ACM CHI 2000.

[5] Buyukkokten, O., Kaljuvee, O., Garcia-Molina, H., Paepcke, A., Winograd, T. *Efficient Web Browsing on Handheld Devices Using Page and Form Summarization*. ACM Transactions on Information Systems, Vol. 20, No. 1, January 2002, p. 82–115.

[6] Gupta, S., Kaiser G., Neistadt D., Grimm, P. *DOM-based content extraction of HTML documents*. WWW 2003: 207 – 214.

[7] GNU Lesser General Public License, http://www.gnu.org/copyleft/lesser.html.

[8] Hoi K.K., Lee D.L., and Xu J. *Document Visualization on Small Displays*. Proc. Int'l Conf. Mobile Data Management, ACM Press, 2003: 262–278.

[9] Fulk, M. *Improving Web Browsing on Handheld Devices*. Proc. ACM CHI 2001.

[10] Furnas, G. *Generalized Fisheye Views*. Proc. ACM CHI 1986, p. 16–23.

[11] Igarashi, T., Hinckley, K. *Speed-dependent Automatic Zooming for Browsing Large Documents*. Proc. ACM UIST 2000, p. 139-148.

[12] Lam H., Baudisch P. *Summary Thumbnails: Readable Overviews for Small Screen Web Browsers*. Proc. ACM CHI 2005, p. 681-690.

[13] Lieberman, H. *A Multiscale, Multilayer Translucent Virtual space*. Proc. IEEE Information Visualization 1997, p. 124-131.

[14] MacKay B. *The Gateway: A Navigation Technique for Migrating to Small Screens*. Proc. ACM CHI 2003.

[15] Milic-Frayling, N., Sommerer, R. *SmartView: Flexible Viewing of Web Page Contents*. Poster paper at WWW 2002 http://www2002.org/CDROM/poster/172/.

[16] Open Mobile Alliance (2001) XHTML Mobile Profile, http://www.openmobilealliance.org/tech/affiliates/wap/wap-277-xhtmlmp-20011029-a.pdf.

[17] Roto, V., Kaikkonen, A. *Perception of Narrow Web Pages on a Mobile Phone*. Proc. Human Factors in Telecommunications 2003.

[18] Roto, V., Popescu, A., Koivisto, A., Vartiainen, E. *Web Page Visualization on Mobile Phones.* Proc. of ACM CHI 2006.

[19] W3C (1998) Cascading Style Sheets (CSS), level http://www.w3.org/TR/REC-CSS2/.

[20] W3C (2000) Document Object Model (DOM) level 2 specification, Version 1. http://www.w3.org/TR/DOM-Level-2-Core/.

[21] W3C (1999) HTML 4.01 Specification, http://www.w3.org/TR/html401.

[22] W3C (2004) Extensible Markup Language (XML) 1.0 (Third Edition) http://www.w3.org/TR/REC-xml/.

[23] Wobbrock J., Forlizzi J., Hudson S., Myers B. *WebThumb: Interaction Techniques for Small-Screen Browsers*. Proc. ACM UIST 2002.

[24] Yang C. and Wang F. *Fractal summarization for mobile devices to access large documents on the web*. WWW 2003: 215-224.

[25] Yin X., Lee W.: *Using Link Analysis to Improve Layout on Mobile Devices*. Proc. WWW 2004.