Pre-print version of the paper. Presented in WWW2008/W4A2008, Beijing, China

MokE: a tool for Mobile-ok evaluation of Web Content

John Garofalakis
University of Patras
Computer Engineering & Informatics Department
26500 Patras, Greece
Computer Technology Institute
26500 Patras, Greece
+302610997866
garofala@ceid.upatras.gr

Vassilios Stefanis
University of Patras
Computer Engineering & Informatics Department
26500 Patras, Greece
Computer Technology Institute
26500 Patras, Greece
+306973281930
stefanis@ceid.upatras.gr

ABSTRACT
The ever-growing corpus of web content is now accessible from mobile devices. But is web content ready for mobile access? Which characteristics provide an acceptable user experience when using a mobile device? Based on W3C's MobileOK best practices and tests we present MokE (Mobile OK Evaluator), a tool that helps creators of web content, webmasters and site administrators to evaluate how a mobile user will experience browsing of their content. By making use of W3C's basic tests this tool is comprised of a set of modules that crawl a web site, analyze its content (including a hidden web capability) and provide an evaluation of its appropriateness for mobile access.

Categories and Subject Descriptors
H.5.2 [**Information interfaces and presentation**]: User Interfaces - Graphical user interfaces (GUI); H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia;

General Terms
Algorithms, Measurement, Design, Experimentation, Human Factors.

Keywords
Mobile web, mobile best practices, mobile OK tests, hidden web, crawlers.

1. INTRODUCTION
The promise of the Mobile Web is a reality. Perhaps not a web suitable for mobile device access but a web accessible by mobile devices. This difference stems from the very nature of unstructured data comprising the web. Web standards have helped to make the web accessible to any personal computer using one from just a handful of the remaining web browsing tools. This does not yet apply to mobile devices. Web content has been build to be accessed from desktop or laptop computers with performance characteristics that are much different from mobile devices: screen size, network bandwidth, support for add-ons/plug-ins, input devices, the limited computational power and memory, the battery issues and energy consumption are some of the characteristics that make the difference. The web is a vehicle for user-centric applications, so accessing content is closely linked to user-friendliness: the user should have a relaxed experience when navigating, usability bottlenecks should be avoided at all costs. In this sense, most of the web is not mobile-friendly.

The most important problem is that users do not have the same navigation experience when they browse the web from mobile devices. This happens because the content is authored without having in mind the limitations of mobile devices [8]. These limitations have led to the creation of the term "Mobile Web". The Mobile Web refers to the World Wide Web as accessed from mobile devices.

The importance of having a real Mobile Web is great since in developing countries the number of users that browse the Web through mobile devices is larger than the number of users that browses the web from desktop PCs. So, mobile Web is providing unique opportunities to bridge the Digital Divide [3]. From the other hand, the goal is not to create a low-level Web for developing countries. Since its creation, the philosophy of W3C is to have one Web ("one world, one web"). If authors of Web content are following (or try to follow) the appropriate guidelines, they can design very rich and attractive documents for users with full browsers (on PC or high-end phones), and the information embedded in those documents will still be available via limited browsers.

The availability of the mobi TLD (dotMobi) a few years ago gave a new perspective to mobile web. Technically, dotMobi is just another domain name, and mobile browsing can use any domain name. So dotMobi is not needed in a technical sense. A dotMobi name is a simple and explicit way to indicate a mobile friendly site. The end-user can choose the site he/she prefers. A dotMobi name also guarantees a reasonable experience for the mobile user. And obviously, the dotMobi domain provides new names for mobile services. So dotMobi is a facilitator for the mobile internet, one of the many components needed to get mobile browsing to the mass market [9]. In contrast to mobile web content, the creation of web content in general is much easier, due to a large number of applications and platforms that promise fast and easy creation without demanding special knowledge from their users. This includes blogs, WIKIs, online web pages creators like Google Page Creator, and WYSIWYG applications like Microsoft's FrontPage, iWeb and Macromedia's Dreamweaver. Users with no programming or design experience have become designers. Web content diversity has increased.

Having this in mind, the question of how to asses the appropriateness of the characteristics of a web site to mobile access requirements arises. Having W3C's Mobile OK guidelines as a starting point, we have developed MokE (Mobile-OK Evaluator), a tool that helps creators of web content, webmasters and site administrators to assess how a mobile user will experience the browsing of their content. By running predefined tests (actually the W3C mobileOK basic tests), against the content's web site the tool provides not only pass/fail results but overall ratings as well.

Based on the W3C's mobileOK checker library, MokE consists of four modules. The first module is a mobile web site crawler that crawls and analyses the given web site. This module acts supplementary with the second module, the hidden web module that discovers hidden content of the given web site. The third module contains the mobileOK checker library and performs the mobileOK basic tests and evaluation based on user-defined weights. Finally, the fourth module is responsible for presenting the system's results to the user and calculating total ratings for the examined web site. The contribution of this work is the exploitation of the hidden content of a web site and the provision of alternative evaluation profiles based on user-oriented significant coefficient weights. The user of the system may analyze the evaluation results in order to evaluate the mobile browsing experience.

This rest of this paper is structured as follows. In section 2 related systems are presented. In section 3 the problem of the hidden web content is described and we define how we approach this problem with our solution. Next, in sections 4 and 5 the W3C Mobile Best Practices and the W3C Mobile OK Tests are presented. Finally, in section 6 we present our approach and in section 7 we conclude.

2. RELATED WORK

Related work in this field includes system and tools that try to examine how web content is presented in a mobile device. One implementation is the W3C Mobile Web Best Practices checker, a tool that takes as input a web page and tests its conformance with some of the W3C Mobile Web Best Practices presented in section 4. This tool was developed as part of the 3GWeb project, financed by the European Commission's IST Program. Now, it has been replaced by [14]. An implementation from Foundation CTIC is presented in [13]. This tool takes as input a URL also but in contrast with the previous tool, tests the URL against mobileOK Basic tests. MobileOK basic tests, presented in section 5, are machine verifiable tests defined from W3C and they are based on W3C mobile web best practices. Furthermore, dotMobi has released a tool which implements many mobileOK Basic tests and performs the tests against a web site providing a summary and a rank for that site [12].

The developers of the above projects are involved in a collaborative effort within W3C Mobile Web Best Practices Working Group to build a reference implementation. The result of these efforts is the mobileOK checker library, where an alpha version was released at October of 2007 by W3C. This library performs the mobileOK Basic tests on a given web resource specified by a URI. A beta web front-end for the library is available from W3C [14]. By creating an open, generic library, W3C enable developers to embed this suite of tests in tools, run a private instance, help enhance and fix the code, or even develop their own new tools.

3. CRAWLERS AND HIDDEN WEB

Crawlers are software that traverse the Web automatically and download pages mainly for use in search engines. In general, a web crawler takes as input a list of URLs. Then the crawler downloads the web content that the URLs point to and by discovering new URLs to that content the procedure continues retroactively [5]. In our case we had to develop a web site crawler. The difference is that a web site crawler takes as an input one URL. This URL is considered as the "base" URL for the web site crawler. During the crawling procedure, every new URL that is discovered is compared to that "base" URL and the crawler adds the new URL to its queue only if the "base" URL is part of the new URL. Before that comparison, the new URL is transformed in the same format with the "base" URL. This transformation includes relative paths for the hyperlinks etc. One problem that may appear with this approach is the redirection to different domains. For example, we assume that the user wants to test the http://www.cisco.mobi web site. When a user visits this URL through a browser, is redirected to a new location, the http://www.cisco.com/web/mobile/. MokE's crawler handles the 302 HTTP redirection. But every new URL will be considered as an external URL compare to the base URL (cisco.mobi) because of the existence of the .com string and the crawling procedure will stop.

Another issue that we had to deal with during the development of the web site crawler is the mobile web orientation of MokE. For example, when a user visits www.google.com from his mobile device he will automatically be redirected to the first page of Google's Mobile version (www.google.com/m). This feature depends on how the remote web server handles the HTTP request headers that the mobile's device browser sends. In any case, MokE's crawler sends the appropriate HTTP headers in order to be recognized as a mobile device from the remote server. The appropriate HTTP request headers are refereed by W3C at the W3C mobileOK Basic Tests document (this issue is presented in detail in section 5).

One main problem of the web site crawler that we had to solve is the "hidden" content of the web sites. The hidden content of a web site is part of the "hidden web" : an ever-increasing amount of information and content on the Web today cannot be reached by following hyperlinks [4]. In particular, a large part of the Web is "hidden" behind forms, and is reachable only when users type in a set of keywords, or queries, to the forms. These pages are often referred to as the Hidden Web [7] or the Deep Web [2], because typical crawlers cannot reach them. The hidden Web has been growing at a very fast pace. It is estimated that there are several million hidden-Web sites [10]. These are sites whose contents typically reside in databases and are only exposed on demand, as users fill out and submit forms. As the volume of hidden information grows, there has been increased interest in techniques that allow users and applications to leverage this information.

The main "entry" to Hidden-Web pages is through querying a form. When the form lists all possible values for a query (e.g., through a drop-down list), the solution is straightforward. A crawler exhaustively issue all possible queries, one query at a time. When the query forms have "free text" input (e.g. textboxes), however, an infinite number of queries are possible and so a crawler cannot exhaustively issue

all possible queries. In this case appropriate queries have to be constructed. Methods for developing crawlers for dealing with this problem are presented in [11] and [1].

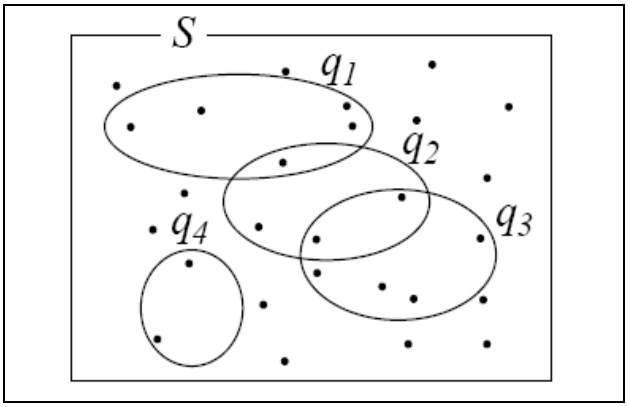

Figure 1. The hidden web problem

If we want to formalize the hidden web site problem, we assume that a crawler downloads pages from a web site that has a set of pages S (the rectangle in Figure 1). We represent each Web page in S as a point (dots in Figure 1). Every potential query q_i that a crawler may issue can be viewed as a subset of S, containing all the points (pages) that are returned when a crawler issues q_i to the site. Under this formalization, the goal of a hidden web crawler is to find which subsets (queries) cover the maximum number of points (Web pages). This problem is equivalent to the set-covering problem in graph theory, an NP-Hard problem [6]. So an efficient algorithm to solve this problem optimally in polynomial time has yet to be found and we have to use approximation approaches.

In MokE we approach the hidden web content of a site from a different point of view. The hidden content of a web site is usually auto-generated. For example, we assume a web site of a library. If a user queries the search form with 20 different words she/he may receive 20 different result pages. But every page will have the same structure in the presentation level. For example, the logo of the library on top, a horizontal line and then the results as a list. So, 20 different queries may return 20 different pages (20 different dots in Figure 1) but from our point of view those queries return one result in the presentation level. Therefore, in our approach we seek the queries q_i that return the most representative hidden web site content in the presentation level. So, in figure 1 with S we represent the whole pages of a web site and with dots the different web content in the level of presentation. Details about the development of this approach and how it works in MokE are presented is section 6.

4. MOBILE WEB AND MOBILE DEVICES

Today, many Web pages are laid out for presentation on desktop size displays, and exploit capabilities of desktop browsing software. Accessing such a Web page on a mobile device often results in a poor or unusable experience. Contributing factors include pages not being laid out as intended. Because of the limited screen size and the limited amount of material that is visible to the user, context and overview are lost. Because of the limited screen size, the subject matter of the page may require considerable scrolling to be visible, especially if the top of the page is occupied by images and navigation links. In these cases the user gets no immediate feedback as to whether their retrieval has resulted in the right content. It is particularly important in the mobile context to help the user create a mental picture of the site. This can be assisted by adopting a consistent style and can be considerably diminished by an uneven style.

Mobile device input is often difficult and certainly very different from a desktop computer equipped with a keyboard. Mobile devices often have only a very limited keypad, with small keys, and sometimes with no pointing device. One of the difficulties of the mobile Web is that URLs are very difficult to type. Lengthy URLs and those that contain a lot of punctuation are particularly difficult to type correctly. Because of the limitations of screen and input, forms are hard to fill in as well. This is because the navigation between fields may not occur in the expected order and because of the difficulty in typing into the fields. While many modern devices provide back buttons, some do not, and in some cases, where back functionality exists, users may not know how to invoke it. This means that it is often very hard to recover from errors, broken links and so on.

Mobile networks can be slow compared to fixed data connections and often have a measurably higher latency. This can lead to long retrieval times, especially for lengthy content and for content that requires a lot of navigation between pages. Mobile data transfer often costs money. The fact that mobile devices frequently support only limited types of content means that a user may follow a link and retrieve information that is unusable on their device. Even if the content type can be interpreted by their device there is often an issue with the experience not being satisfactory - for example, larger images may only be viewable in small pieces and require considerable scrolling. Web pages can contain content that the user has not specifically requested for - especially advertising-related images or large images. In the mobile world this extra material contributes to poor usability and may add considerably to the cost of the retrieval.

Mobile users typically have different interests to users of fixed or desktop devices. They are likely to have more immediate and goal-directed intentions than desktop Web users. Their intentions are often to find out specific pieces of information that are relevant to their context. An example of such a goal-directed application might be a user requiring specific information about schedules for a journey he/she is currently undertaking. Mobile users are typically less interested in lengthy documents or in browsing lengthy pages. The ergonomics of

the device are frequently unsuitable for reading lengthy documents, and users will often only access such information from mobile devices only when more convenient access is not available.

Developers of commercial Web sites should have in mind that different commercial models are often at work when the Web is accessed from mobile devices as compared with desktop devices. For example, some mechanisms that are commonly used for presentation of advertising material (such as pop-ups, pop-unders and large banners) do not work well on small devices. It is not the intention of the MWI to limit or to restrict advertising; rather it is the intention that the user experience of the site as a whole, including advertising is as effective as possible.

As noted above, the restrictions imposed by the keyboard and the screen typically require a different approach to page design than for desktop devices. Various other limitations may apply and these have an impact on the usability of the Web from a mobile device. Mobile browsers often do not support scripting or plug-ins, which means that the range of content that they support is limited. In many cases the user has no choice of browser and upgrading is not possible. Some activities associated with rendering Web pages are computationally intensive - for example re-flowing pages, laying out tables, processing unnecessarily long and complex style sheets and handling invalid markup. Mobile devices typically have quite limited processing power which means that page rendering may take a noticeable time to complete. As well as introducing a noticeable delay, such processing uses more power as does communication with the server. Many devices have limited memory available for pages and images, and exceeding their memory limitations results in incomplete display and can cause other problems.

5. MOBILE WEB BEST PRACTICES

In this section the W3C mobile web best practices document is presented in brief. The full document may be found in [15]. Mobile web best practices and mobile ok basic tests that are presented in the next section, are the result of two different working groups of W3C Mobile Web Initiative (MWI). The Mobile Web Initiative is led by worldwide key players in the mobile production chain, including authoring tool vendors, content providers, adaptation providers, handset manufacturers, browser vendors and mobile operators. There are nineteen MWI Sponsors: Ericsson, France Telecom, HP, Nokia, NTT DoCoMo, TIM Italia, Vodafone Group Services Limited, Afilias, Bango, Jataayu Software, Mobileaware Ltd., Opera Software, Segala, Sevenval AG, Rulespace and Volantis Systems Ltd.

The mobile web best practices document specifies best practices for delivering Web content to mobile devices. The principal objective is to improve the user experience of the Web when accessed from such devices. The recommendations refer to delivered content and not to the processes by which it is created, nor to the devices or user agents to which it is delivered (Figure 2). In other words, mobile web best practices refer to how the web content should be presented to the end user, independently to his/her device or to the adaptation mechanisms the network may use (e.g. content adaptation proxies).

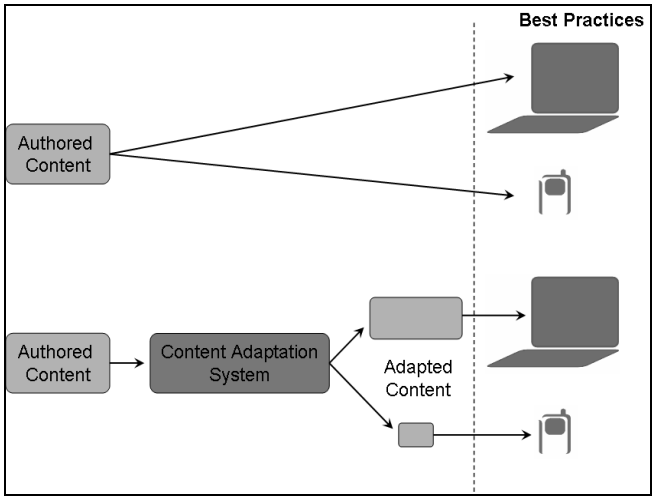

Figure 2. Delivering content.

The sixty best practice statements are grouped in five categories a) Overall Behavior: General principles that underlie delivery to mobile devices b) Navigation and Links: Because of the limitations in display and of input mechanisms, the possible absence of a pointing device and other constraints of mobile devices, care should be exercised in defining the structure and the navigation model of a Web site c) Page Layout and Content: This category refers to the user's perception of the delivered content. It concentrates on design, the language used in its text and the spatial relationship between constituent components. It does not address the technical aspects of how the delivered content is constructed d) Page Definition and e) User Input: This section contains statements relating to user input. This is typically more restrictive on mobile devices than on desktop computers and often a lot more restrictive.

In order to allow content providers to share a consistent view of a default mobile experience the W3C has defined the Default Delivery Context, a simple and largely hypothetical mobile user agent. This allows providers to create appropriate experiences in the absence of adaptation and provides a baseline experience where adaptation is used. The Default Delivery Context (DDC) has been determined by the W3C as being the minimum delivery context specification necessary for a reasonable experience of the Web. It is recognized that devices

that do not meet this specification can provide a reasonable experience of other non-Web services. The Default Delivery Context is defined in Table 1.

Table 1. Default Delivery Context

Characteristic	Value
Usable Screen Width	120 pixels, minimum
Markup Language Support	XHTML Basic 1.1 delivered with content type application/ xhtml+xml
Character Encoding	UTF-8
Image Format Support	JPEG and GIF 89a
Maximum Total Page Weight	20 kilobytes
Colors	256 Colors, minimum
Style Sheet Support	CSS Level 1. In addition, CSS Level 2@media rule together with the handheld and all media types
HTTP	HTTP ver1.0 or more recent HTTP ver1.1
Script	No support for client side scripting

It must be noted that many devices exceed the capabilities defined by the DDC. Content providers are encouraged not to diminish the user experience on those devices by developing only to the DDC specification, and are encouraged to adapt their content, where appropriate, to exploit the capabilities of the actual device.

6. MOBILE-OK TESTS

In this section the W3C mobileOK basic tests are presented. The full document may be found in [16]. MobileOK Basic tests are based on a limited subset of the Mobile Web Best Practices. MobileOK Basic tests is a scheme for assessing whether web content can be delivered in a manner that is conformant with Mobile Web Best Practices to a simple and largely hypothetical mobile user agent. This mobile user agent is the Default Delivery Context (DDC) - presented in table 1. The outcome of the mobileOK basic tests is machine verifiable; hence claims of mobileOK Basic conformance can be checked automatically. Content passing the tests demonstrates that the creator of the content has taken basic steps to provide a functional experience for mobile users.

Furthermore, W3C refers to mobileOK Pro tests too. The mobileOK Pro tests include the mobileOK Basic tests and are based on a larger subset of the Mobile Web Best Practices. These tests are not all machine-verifiable. Content passing the tests demonstrates that the content provider has also taken significant steps to provide a functional experience for mobile users. A first version of a document that describes the pro tests is expected to be public available from W3C in late 2008.

In the following paragraphs the mobileOK basic tests are presented. The name of each test is the same as this is generated from the W3C mobile checker library that is presented in subsection 6.2. W3C does not prescribe the order in which mobileOK basic tests are to be carried out and they may be executed independently. So, the tests are presented in alphabetic order.

AUTO_REFRESH_REDIRECTION: Tests if auto-refresh mechanism is presented or redirections different than the HTTP headers redirections are used. This test does not determine whether the user is able to opt out of refresh.

CACHING: Tests the HTTP headers for caching information. The purpose of the test is to alert the user to the fact that his/her content may not be cached.

CHARACTER_ENCODING_SUPPORT: The DDC is defined to support only UTF-8 encoding, and hence this test fails if a resource is not encoded in UTF-8. The test does not require that resource always be encoded in UTF-8; the test merely checks that the resource is available in UTF-8 encoding, if requested.

CONTENT_FORMAT_SUPPORT_AND_VALID_MARKUP: Tests if the content is send in a format that is known to be supported by the device. This is information can be extracted from HTTP User-Agent header, HTTP Accept headers and UAProf. Also the validity of the mark up against published formal grammars is tested.

DEFAULT_INPUT_MODE: Tests if a default text entry mode, language and/or input format is specified in the forms.

EXTERNAL_RESOURCES: Make an inventory of unique included resources (images, style sheets and other objects). If the total exceeds 10, the test returns WARN. If this total exceeds 20, returns FAIL.

GRAPHICS_FOR_SPACING: Best Practices advice to avoid using transparent images for spacing. Therefore this test merely warns about the presence of small (at most 2x2) transparent images and fails larger ones.

IMAGE_MAPS: Check for the existence of image maps.

IMAGES_RESIZING_IMAGES_SPECIFY_SIZE: Checks if the size of images in markup is specified.

LARGE_GRAPHICS: Tests the size of the images.

LINK_TARGET_FORMAT: Checks if the target file's format for each linked resource is supported from the DDC.

MEASURES: Tests if pixel measures are used in markup language attribute values and style sheet property values.

MINIMIZE: Counts the number of white spaces in a sequence in source code and compares the sum with the total size of the source.

NO_FRAMES: Tests if the content contains frames.

NON_TEXT_ALTERNATIVES: Tests if a text equivalent for every non-text element is provided.

OBJECTS_OR_SCRIPT: Checks for the existence of scripts or other objects that are not supported from the DDC.

PAGE_SIZE_LIMIT: Tests if the total size of the content exceeds the limit of the DDC.

PAGE_TITLE: Checks if the page has the title tag into the head.

POP_UPS: Tests if the source code may generate pop up windows.

PROVIDE_DEFAULTS: Tests if pre-selected default values are provided in forms elements like radio buttons and drop down menus.

SCROLLING: Tests if the content will require horizontal scrolling when presented in the DDC.

STYLE_SHEETS_SUPPORT and STYLE_SHEETS_USE: Checks the CSS for certain values and attributes.

TABLES_ALTERNATIVES: Checks for the existence of html tables.

TABLES_LAYOUT: Checks if the html tables are used for layout purposes.

TABLES_NESTED: Tests if a the source code contains nested tables.

7. THE MOBILE-ok EVALUATOR

MokE is a web based application and so the only requirement for the end user is a web browser. The server that hosts the application requires the Apache Web Server version 2, PHP version 5 and Java runtime environment 1.5 or higher. We are going to present the steps of using the tool by means of a usage scenario. We assume that a new user wishes to test his/her site. The steps that will be followed are a) Creation of a new crawler for the site b) Checking for hidden web content in the site c) Run of the MobileOK Basic tests against the web content that gathered from steps a and b. d) Evaluating the web site based on the results of step c and the weighted profiles created by the user.

First, MokE requires from the user to log in. So, the new user creates a new account. When the user is identified, the main screen for the registered user appears (figure 3). The screen is divided in two areas. The left part contains the menu of options for the user and the right part is where the feedback appears. All the following screenshots are mentioned to that right area that is the only one that changes.

Figure 3. Main page of our system.

7.1 Mobile Crawler

The user, after his/her login, has to create a new crawler for her/his site. By choosing “new crawler” from the menu the form depicted in figure 4 appears.

The user has to enter the URL of the site that she/he wants to crawl. This URL may refer to any site, not only to web sites that are designed for mobile devices. In any case, the procedure will continue but the rating of the site will be possible poor. The other field that is required is “Depth”. Here the user defines the depth of the crawling. The URL that the user typed is level 0. This URL is also considered as the

"base" URL for deciding if a new URL belongs to the web site or not, as described in section 3. The user may give any value to the field "Depth". However, our tests show that for values over 5 a lot of time and computational power is needed for the crawling procedure.

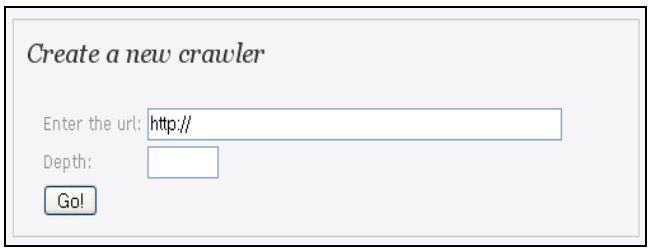

Figure 4. New crawler.

The crawler has to send proper HTTP request headers to inform the server that it should deliver content that is compatible with the Default Delivery Context, if this is available. The user-agent header has to be "User-Agent: W3C-mobileOK/DDC-1.0 ", the accept header has to contain content types that the Default Delivery Context accepts and, finally, the accept-charset header has to be "Accept-Charset: UTF-8".

When the crawling procedure is finished the user is informed for the number of pages that have been found, the number of the new site URLs that every page contained and for the total data that have been transferred and saved to the database. Then, the user may continue to step 2 where the mobileOK checker module tests every page that the crawler discovered for conformance with the mobileOK basic tests.

7.2 MobileOK Checker

The core of the MobileOK checker module is the mobileOK Checker of W3C. The library is implemented in Java and can be called from the command line. The library gets the following input: a) URL: The web resource to check against mobile OK Basic Tests. For example http://www.example.org. b) Tests: An optional argument to check the web resource against a list of mobileOK Basic Test. By default, if this argument is not present, all mobileOK Basic Tests are checked. The name of each test is the same the ones shown in Table 1. c) Proxy: DNS name or IP and port number of an HTTP intermediate proxy. d) Results file: Results are saved in the specified file. e) Intermediate file: The mobileOK Intermediate document (mOKI) is saved in the specified file.

When tests are performed by the W3C mobileOK Checker on a web resource, two XML documents are generated. After retrieving the web resource to be examined, a preprocessing stage is performed by the Checker library and the first XML document is generated, the mOKI. This document is then processed in order to generate the final results document, where the output of the tests is stated. The mOKI document contains the HTTP dialog taking place between the Checker library and the remote server in order to retrieve the web resource, including HTTP requests, HTTP responses, the content of the web resource as delivered by the server in one of the responses and, of course, all the HTTP headers included in requests/responses. It also contains information about the validity of the source code of the document (XHTML valid, UTF-8 valid) and information about external resources of web page like images and CSS documents. All this information is really important for the MobileOK basic tests that will follow in order the final XML file with results to be generated.

The final document of W3C mobileOK Checker library is generated after processing the previously mentioned mOKI document. It consists of an XML root element named tests which contains a sequence of test elements. Each test element will be qualified by an outcome attribute. The value of outcome can be "FAIL", "Pass" or "WARN" and refers to the outcome of the correspondent MobileOK Basic Test as they described in section 5. In figure 5 a part of a sample XML file with results is presented.

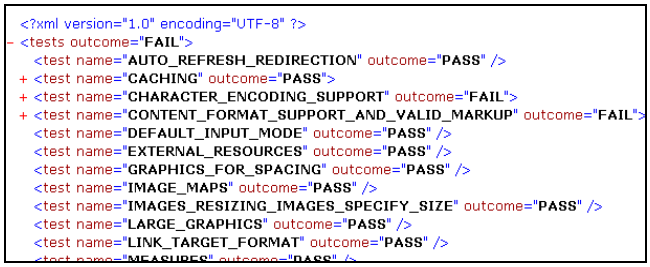

Figure 5. Results file.

Finally, the system informs the user for the end of the procedure and the mobileOK checker module reads the results for every web page from the XML files and stores the data to a MySQL database. Those data will be used from another module of the system, described later in the section, for calculating the ratings for the site.

7.3 Hidden Web Module

As mentioned in section 3, forms are considered as the main data entry method to hidden web content. Therefore, the hidden web module has to search for html forms in the crawled data. To make this possible, during the crawling procedure, MokE stores to its database the source code of every crawled web page, besides its URL. The user has to follow two steps in order to use this module. The first step is a search for hidden content. The second uses the mobileOK checker module to test the hidden content. The results of those tests are taking into account for calculating the rating of the site.

In the first step the module searches to the database in order to find the form html element in the pages that have been crawled for the current site. Then all the forms are presented to the user. An example is presented in figure 6.

The presentation of the form is the same as the presentation of the form at the original page. Also, the user may click to the URL of the page that the form has been found in order to access in a new window the original page that contains the form and may type more suitable keywords in the fields. The page "page2.html" that contains the form of figure 6 is presented in figure 7.

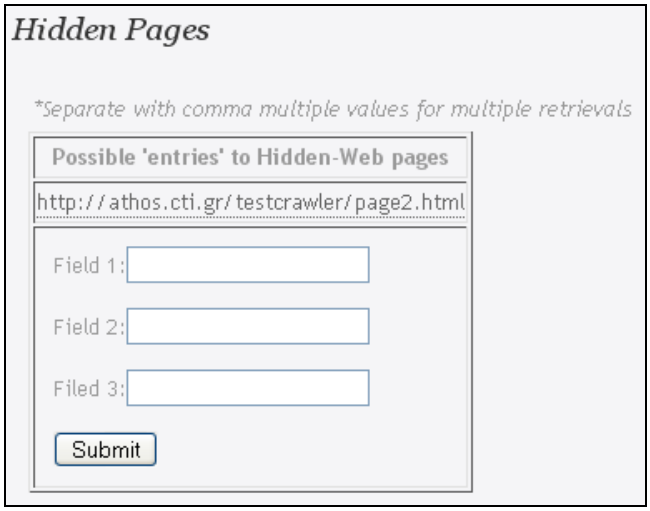

Figure 6. Hidden web module.

The form that is presented to the user is the same as the original form. The difference is that the rest of the page (the links on top, the heading and the links after the form) are omitted. The user may fill in the fields with values and push the "Submit" button.

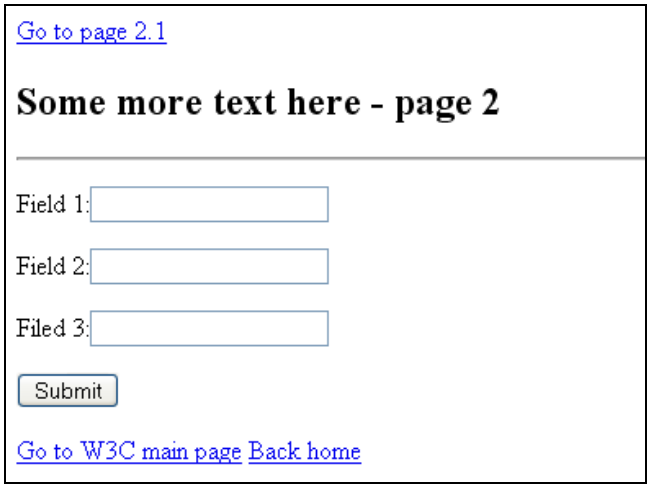

Figure 7. Hidden web example.

The hidden web module changes the field action of the form so all data (including hidden values) are sent back to the hidden web module. Then, the module is responsible to send the form's request to the original target and retrieve the hidden content. Afterwards, the mobileOK module runs all the mobileOK basic tests against the hidden content.

In order to minimize the time that the system requires to fetch a part of the hidden content the user has the option to fill in the fields of the form with multiple values separated by commas. For example, in figure 6 the user may type six words in every field separated by a comma. A matrix of 3X6 is constructed. When the "Submit" button is pressed the module sents six different requests to the original target of the form. Each request contains one column of the 3X6 matrix. Also, if the form contains drop down menus, checkboxes and radio buttons the user may make selections for those fields too. Those values and the values from hidden variables are sent in every request to the remote server. So, in the above example, with one submit the module retrieves six different instances of hidden web content generated from one form. It is obvious that this procedure causes the minimum load to the remote server and to MokE. Because of the user's assistance and the auto generated nature of the hidden web the hidden content that is discovered is representative of the site in the presentation level.

7.4 Web Site Rating

When the user has finished with the crawling and the hidden content of the web site she/he may view the results and the total rating of the site (Figure 8). The result page is divided in four areas. In the first area the user may see a summary of the number of pages that have return

PASS, FAIL or WARN in each of the twenty six mobileOK basic tests. For example, for the test TABLES_NESTED the system presents "TABLES_NESTED Pass: 4/6, Fail: 2/6, Warning: 0/6". This means that four (4) of the crawled pages have passed the specific test and two (2) have failed. In the next area the user gets the same information about the content that was discovered from the hidden web module. This information is presented separately to the user in order to evaluate how his/her site presents the hidden content. Next, the third area contains briefly the results of the two previous areas in percentage. For example, figure 8 presents that for the eight (8) crawler web pages of the web site, 208 tests have run by our system (8 pages * 26 mobileOK tests). 182 tests out of 208 (87.5%) returned PASS, 25 returned FAIL (12.02%) and 1 returned WARN (0.48%).

Finally the fourth area contains two numbers. The first number is the evaluation rating R of the site and is calculated as follows:

$$R = \frac{\frac{10}{P}\sum_{1}^{n}(P_{pass}^{tn} + 0.5 * P_{warn}^{tn})}{n}$$

Where P is the total number of the pages tested, P_{pass}^{tn} is the number of pages that returned PASS in the test t_n, P_{warn}^{tn} is the number of pages that returned WANR in the test t_n and **n** is the number of tests (26 in our case). The calculation of R is based on a three level scale that reflects the impact of PASS, WARN and FAIL to the total evaluation rating. The significant coefficient of PASS result is defined as 1, FAIL is defined as 0 and WARN is defined as 0.5. Finally, we multiply with 10/P in order to normalize the evaluation rating in the scale 0 – 10. So, R provides a total evaluation rate for the web site based on the results of the mobileOK basic tests for every web page of the site.

The system reasons how and which mobileOK basic tests should be used in different evaluation scenarios. Since a consensus on the significance (how good is a mobileOK test in the evaluation process) of mobileOK tests presented is subjective (i.e used dependent) we have foreseen the inclusion of significance weight in the system. The user creates profiles using weights of importance for every mobileOK basic test. The weight for every test is a value using a five-grade Liker-type scale of (1 = not important, 2 important, 3 = average important, 4 = very important, 5 = critical). The weight is set by default to 5 for all mobileOK tests in order to denote their equal importance. The user may also select 0 if he/she wants to ignore the results of a specific test. In that case the weighted evaluation rating WR of the web site is calculated by the formula:

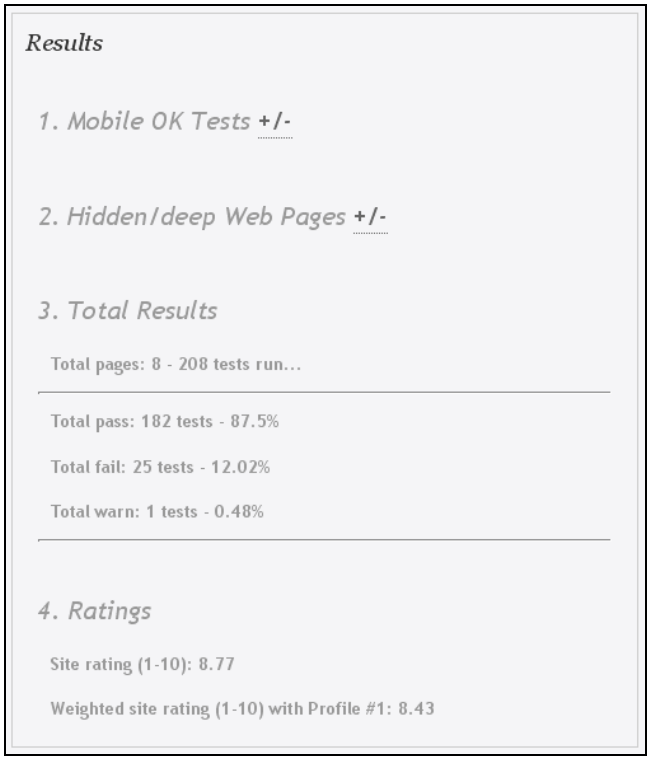

Figure 8. Results page.

Where W_n is the weight that the user has specified for each test, while a zero value ignores the test. MokE allows the user to create any number of alternative weighted profiles in order to compare how the weighted evaluation rating of the site changes. The WR parameter denotes the importance of every mobileOK basic test from the user's perspective. This makes the system flexible by attacking the problem of subjectivity in the mobileOK evaluation. Thus different users may operate on different instances of the system by increasing or decreasing the weights of tests depending on their perception of web content.

$$WR = \frac{\frac{10}{P}\sum_{1}^{n} w_n(P_{pass}^{tn} + 0.5 * P_{warn}^{tn})}{\sum_{1}^{n} w_n}$$

8. FUTURE WORK

In this paper we presented MokE, a tool for rating web content as presented when accessed from mobile devices. Our system crawls and tests a part of the hidden content of a web site, in order to calculate the final rating. Apart from the rating of the content a summary with the results of the mobileOK basic tests for the whole content is presented.

A new module that will produce recommendations for the user in order to improve the achieved rating of his/her content is necessary. Also, the hidden content of site may contain new hyperlinks and new forms that are entries to other content of the site. So, an improved crawler that will continue the crawling procedure retrospectively to hidden content has to be developed. Finally, experiments with a large number of web sites have to be carried out in order to measure the impact of the hidden content to the rating of the whole web site. All the above matters will be included in our future work.

9. REFERENCES

[1] Barbosa, L., and Freire, J., An Adaptive Crawler for Locating HiddenWeb Entry Points, In Proc. of WWW 2007.

[2] Bergman. M., The Deep Web: Surfacing Hidden Value, http://www.press.umich.edu/jep/07-01/bergman.html, 2001.

[3] Boyera, S., The Mobile Web to bridge the Digital Divide, Volume 14/Issue 3, ACM Interaction Magazine, 2007, pp. 12-14.

[4] Chang, K., He, B., Li, C., and Zhang, Z., Structured Databases on the Web: Observations and Implications, Technical report, UIUC, 2003.

[5] Cho, J., and Molina, H., Parallel Crawlers, In Proc. of WWW 2002.

[6] Cormen, T., Leiserson, C., and Rivest, R., Introduction to Algorithms, 2nd Edition. MIT Press/McGraw Hill, 2001.

[7] Florescu, D., Levy, A., and Mendelzon. A., Database Techniques for the World Wide Web: A Survey, ACM SIGMOD Record, Volume 27, Issue 3, p.p. 59 - 74, ACM, 1998.

[8] Garofalakis, J., and Stefanis, V., Using RSS feeds for effective mobile web browsing, Universal Access in the Information Society Journal, Volume 6, Number 3 /November, p.p. 249-257, Springer, 2007.

[9] Haumont,S., and Siren, R., dotMobi, a Key Enabler for the Mobile Internet, In Proc. of Workshop on Mobile Internet User Experience, Mobile HCI 2007.

[10] Hsieh, W., Madhavan, J., and Pike, R., Data management projects at Google, In Proc. of ACM SIGMOD, pages 725–726, 2006.

[11] Ntoulas, A., Zerfos, P., and Cho, J., Downloading Textual Hidden Web Content Through Keyword Queries, In Proc. of JCDL 2005.

[12] Ready.mobi site check, http://ready.mobi

[13] TAW mobileOK Basic checker, http://validadores.tawdis.net/mobileok/en/

[14] W3C mobileOK Checker Beta release, http://validator.w3.org/mobile

[15] W3C, Mobile Web Best Practices 1.0, http://www.w3.org/TR/mobile-bp/

[16] W3C, MobileOK Basic Tests 1.0, http://www.w3.org/TR/mobileOK-basic10-tests/