A RESTful web service to estimating time requirements for web forms

Ludger Martin

Hochschule RheinMain,
Faculty of Design – Computer Science – Media,
University of Applied Sciences,
Unter den Eichen 5,
65195 Wiesbaden, Germany
E-mail: ludger.martin@hs-rm.de

Abstract: It is important to know whether a website or web application is usable. What you need are statistics about how the website is used. Such statistics usually include information about how much time a web user spends on a web page. But to know whether those dwell times are acceptable you will need reference values. This paper provides and discusses such reference values based on a research study. To allow automatic analysis of input elements a web service is presented to reveal any possible element. Using this service and the results of our study, any kind of HTML/HTML5 input elements in a web form can be analysed.

Keywords: web usability; web forms; HTML input elements; RESTful web services; knowledge extraction.

Reference to this paper should be made as follows: Martin, L. (2014) 'A RESTful web service to estimating time requirements for web forms', *Int. J. Knowledge and Web Intelligence*, Vol. 5, No. 1, pp.62–75.

Biographical notes: Ludger Martin is a Professor in the Faculty of Design – Computer Science – Media, at the University of Applied Sciences, Wiesbaden Rüsselsheim, Germany. His research interests include web usability, semantic web and data mining.

This paper is a revised and expanded version of a paper entitled 'Estimating time requirements for web input elements' presented at 2nd International Workshop on Web Intelligence, Angers, France, July 2013.

1 Introduction

Many websites on the internet gather various statistics on their usage. Two of the criteria that are collected are the time a web user stays on a particular web page (*dwell time*) and information on which web pages the user visits. Analysis of the collected data can provide an indication on whether the website is well designed or not. To perform this analysis, it is essential to know whether particular dwell times should be considered good or bad.

This paper aims to provide criteria for more precision in assessing dwell times by examining HTML and HTML5 input elements. A survey shall help to estimate the time it will take a user to fill in a form on a web page. Statistics about the various types of input elements will be created. It is our aim to enable an automatic estimation of the average time it will take users to fill in any web form. Therefore a RESTful web service is presented. This web service determines all input elements on a single web page. Using this data, possible dwell times for any web form can be calculated.

The time required for filling in forms can then be supplemented with further values obtained about a web page. For example, the quantity, the readability and the main subject of a text, as presented in Martin (2010, 2012). All of this information as a whole will help to understand the statistics about a website. The main goal is to find automatically problems in the usability of a web page.

The remainder of this paper is structured as follows: in Section 2 the study and the questionnaire used will be presented. Then, in Section 3 the results of the study are discussed. Section 4 demonstrates the application of the results related to a real form. A web service to automate this process is introduced in Section 5. After presenting related work in Section 6 we conclude the paper with an outlook on future work.

2 Survey questionnaire

An online questionnaire has been compiled for this study. Test persons are asked to input their answers to the various questions into the questionnaire. While they are filling in the form, a JavaScript records the time the test persons spent on each question. To make the time measurement more informative, the questions that are not currently dealt with are greyed out. This method ensures that the test persons can read and answer only one question at a time. It is the test persons' decision whether they want to use a desktop browser or a mobile browser. Environmental factors such as where and under which circumstances the test persons access the website, were not analysed.

The questions do not require much contemplating, e.g., the first question asks the users to describe what they did after getting up in the morning. This is followed by a choice of seven radio buttons about how the test persons came to work that day. Then they are asked to enter their favourite food in a single-line text field. Additionally, statistical data is maintained about the person's gender using three radio buttons. The next question asks for another piece of text. This time, a word from the title of a book is to be entered. As a rule, the type of input element is never the same as the one directly before. The next four questions use check boxes. A further statistical question has the form of a select field. Here, age ranges are chosen. Six lines are displayed at the same time. If users want to enter an age older than 34, they must scroll. The effect of scrolling will be described in greater detail later on. In the next field, the date of the day before is required. If the browser supports HTML5, the appropriate HTML5 date field will be displayed, if not, a simple text field. In our analysis we will look at these differences in greater detail. If the browser supports them, the colour field and the range field will also be displayed. The form is completed with a simple submit button.

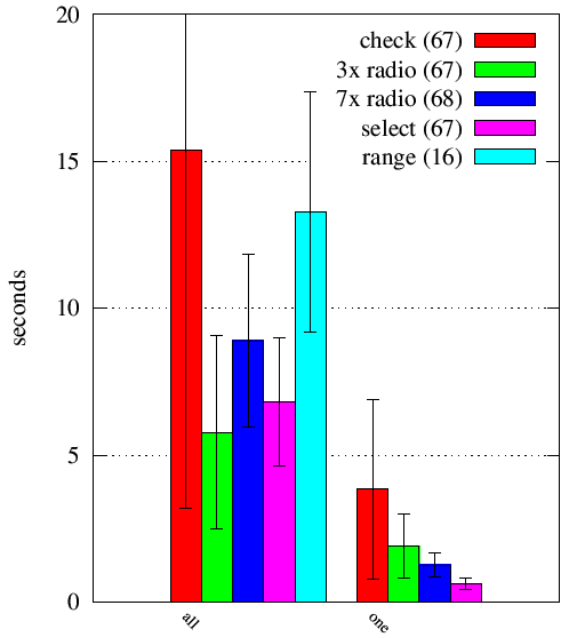
The other new HTML5-elements such as number, url or e-mail are just short text fields and are therefore not considered separately in this study.

3 Analysis

In January and February, 2013, a total of 80 test persons participated in the study. The diagrams below show in round brackets the number of test persons who have filled in a particular input field. The differing numbers are due to the fact that not all test persons filled in all of the input fields and, depending on the browser, not all of the HTML5 input elements were displayed. In the following examinations only the values of the test persons who used a desktop browser were analysed.

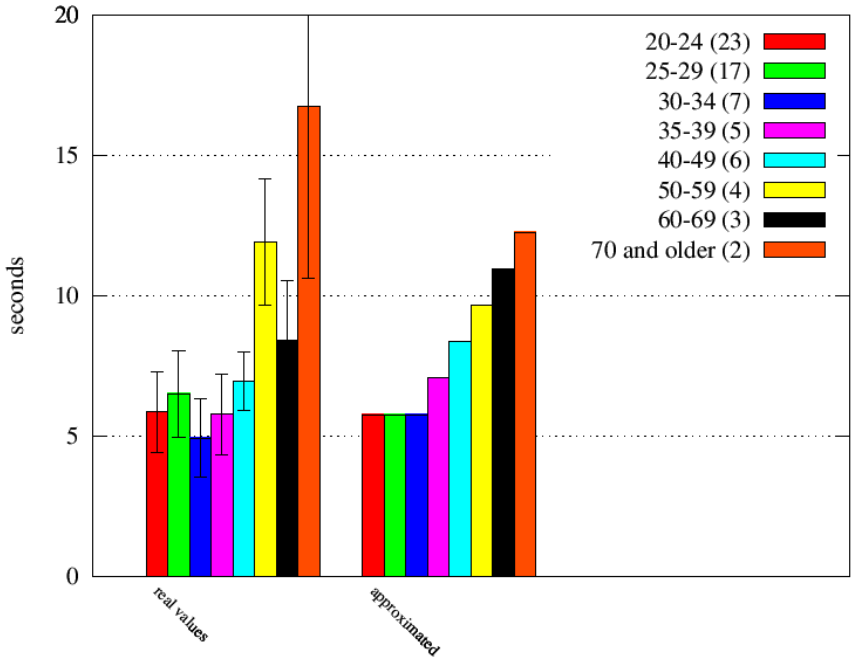
Figure 1 shows the input elements which can be selected using the mouse or the keyboard. The four check boxes together required an average time of 15.4 s. Compared with the three radio buttons about gender, we note that these could be answered much faster (5.8 s). Even the seven radio buttons took less time. The reason for this is that each check box requires an individual answer while the radio buttons belong together. But the time required to deal with a group of radio buttons depends on the number of group elements. The select field for the person's age is similar to the radio buttons, as it required a similar amount of time. Entirely different is the range field. Here, a user is to choose a predefined value. This field seems to pose most of the problems as it requires an average time of 13.3 s. To facilitate applying the characteristic numbers to a form you want to analyse, we have recalculated the numbers for the individual elements also. For example, it takes 3.0 s to deal with a check box. Figure 1 also shows that the standard deviation varies considerably. We will look at this in greater detail later in this section.

Figure 1 Selection input elements (see online version for colours)



Now we will take a closer look at the select field. Figure 2 relates the time with the options chosen. At the same time, the options represent the test persons' ages. The first options show similar times and standard deviations. Only starting with option '35–39' does it become necessary to scroll down. This shows that if more scrolling was necessary the required time increased considerably. The increase in time might be explained by the higher age of the test person. But admittedly not many senior persons participated in the study and therefore these values must be used with caution.

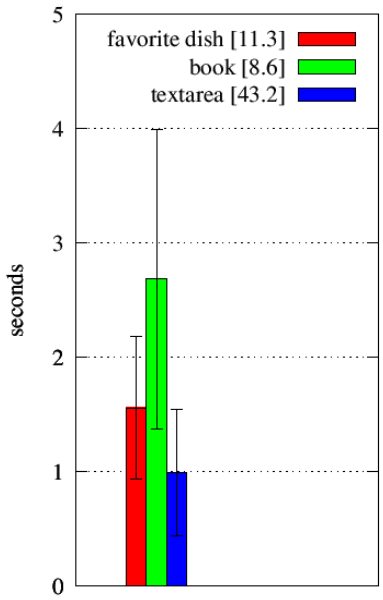
Figure 2 Selection and scrolling (see online version for colours)



In the second part of Figure 2 we have tried to create a formula for select field usage. For choosing an option without scrolling we assumed 5.6 s. For each line that must be scrolled we have added 2.3 s. Figure 2 shows this using the age list as an example, where quite a good approximation can be seen. We have allowed extra time for the standard deviations as well.

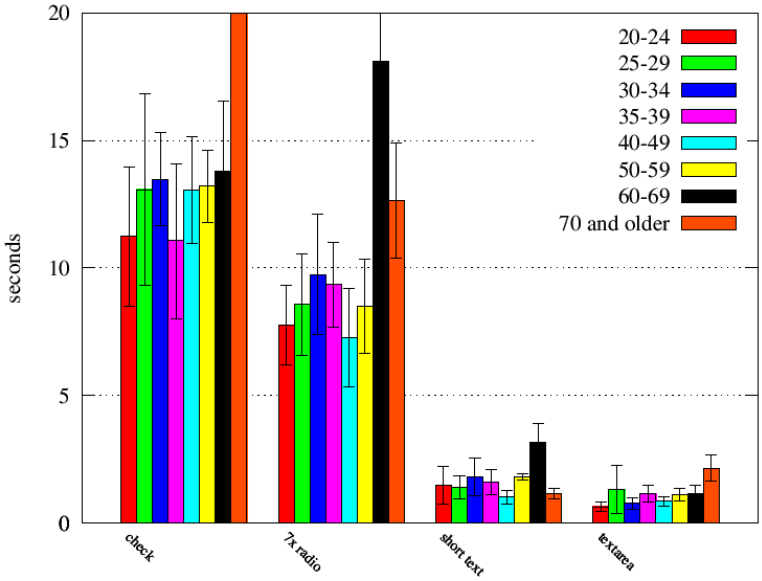
Next we will look at the three elements which require text entry using the keyboard. The date field is left out as no normal text is entered there. The angle brackets in Figure 3 specify the average numbers of letters that were input. The times refer to one character each. Interestingly, the question about a person's favourite food was answered more quickly than the question about a word from the title of a book. The test persons obviously thought longer about the second question. Another observation was that longer pieces of text can be entered significantly quicker.

Figure 3 Text input elements (see online version for colours)



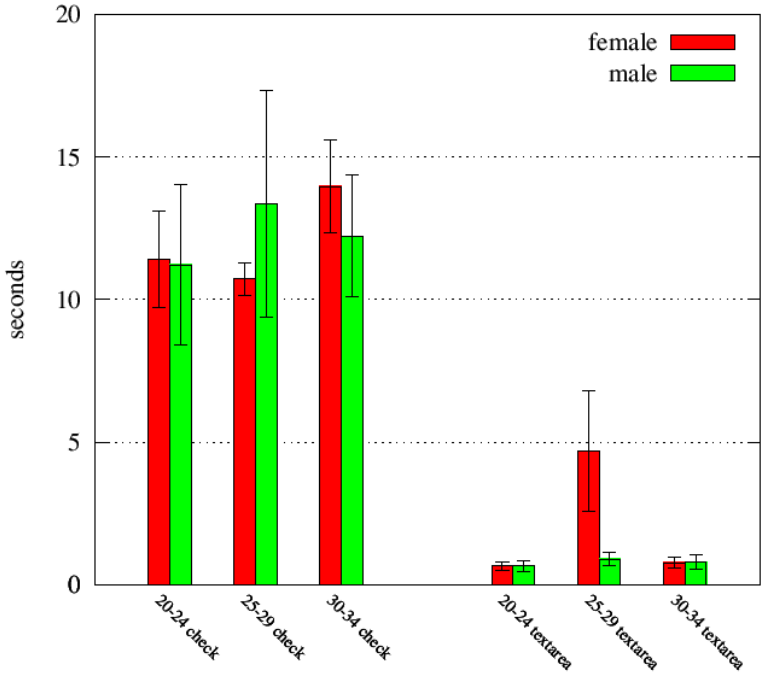
Before that, larger standard deviations had been noted. These are now compared to the test persons' ages. Figure 4 shows minor standard deviations within the age ranges. The larger values were created by the senior test persons (60 or 70 and older) as also observed in Figure 2. If you know the age of a website's target group, then only the average values for this age group are to be used. This paper continues to look at all age groups.

Figure 4 Effect of age (see online version for colours)



After examining the age, now we want to look at the gender. In addition to the check boxes and the text area, Figure 5 lists three age groups with the majority of test persons. Notably, with the female test persons there were fewer standard deviations. This may be due to the fact that fewer women participated in the survey. Hardly any difference can be seen between the female and male test persons at least in the age groups '20–24' and '30–34'. The big difference in the text area of the age group '25–29' may be explained by the small number of test persons.

Figure 5 Effect of gender (see online version for colours)



One novelty in HTML5 standard is, among others, an input field for a date. In Figure 6, this date field is compared to a field where the date must be typed in. The date field includes a calendar from which the date can be chosen. With two clicks you can select a date. In a text field, however, ten characters must be entered. Figure 6 shows this extra effort. Date fields are about double as efficient as a text field where the date must be entered. Another new HTML5 is the colour field. An average time of 13.5 s could be determined for that field. However, only three people filled in this field.

So far we have only evaluated data for desktop browsers. Figure 7 compares the test persons who used a mobile browser to those who used a desktop browser. Out of the 80 test persons, 12 persons used mobile devices. Out of these, seven persons used smart phones. In a test prior to this study, the presentation of the form on desktop browsers was compared to the presentation on mobile browsers. The elements were arranged identically. Just the size of the form itself was different. For date fields or colour fields no data was available. Apart from the range field there were no big differences. The range field had already been different in desktop browsers and this proved even more true using

mobile browsers. This makes us question the usability of the range field. Text fields hardly showed any differences. With longer texts the time spent per character was slightly more. This can be explained by the fact that typing on mobile devices is less convenient.

Figure 6 Comparison of date and text input (see online version for colours)

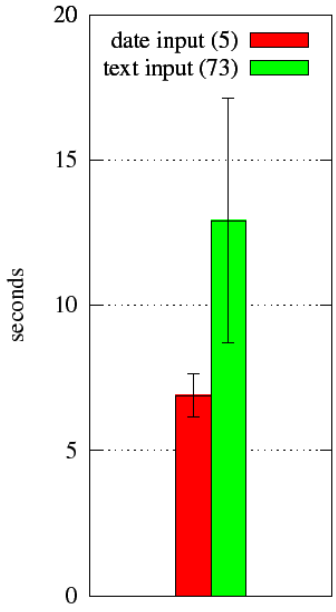
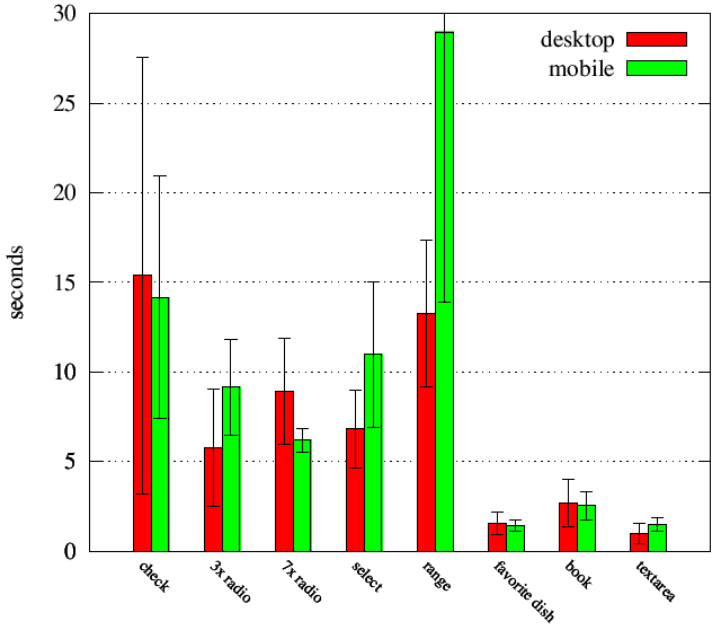


Figure 7 Comparison desktop/mobile (see online version for colours)



The form is sent off using the submit button. We did not examine these buttons as they are only pressed once to submit the form. Measuring time is hardly possible, especially as every form is submitted using this button and users do not spend much time on this kind of button.

Table 1 shows a summary of all times we ascertained. We calculated only one value for all of the check boxes and one for all of the radio buttons. For mobile browsers we could not determine values from scrolling a select field, as too few participants were available. For the texts, as before, we only specified the input times for one character. As mentioned before, we did not investigate the environmental factors of the test persons. In general, when analysing web statistics it is not possible to detect the users environment. The times shown in Table 1 shall be used to estimate the time it will probably take to fill in an entire form.

Table 1 Selection and scrolling

	Desktop	*Mobile*
Check	3.84 s	3.54 s
Radio	1.60 s	1.97 s
Select	5.76 s	10.60 s
Select scroll	1.30 s	-
Range	13.28 s	28.96 s
Short text	2.12 s	1.97 s
Text area	0.99 s	1.49 s
Date	6.90 s	-
Colour	13.50 s	-

4 Case study

In this case study, an order form of a small web page is analysed. The input fields of the form are shown in Table 2. Some of the input fields are optional, some required. The specified times represent users of a desktop browser and are calculated related to Table 1. The calculation related to short text fields observe the following rules:

- If the attributes `size` and `maxlength` are equal, it will be assumed that exactly `size` character need to be specified.
- If the attributes `size` and `maxlength` are different, it will be assumed that the size of the input field is as big as the most common entries. But the entries might vary in length. Therefore, the calculation uses 50% of the characters specified in the `size` attribute.

In our study the test persons (see three-typed an average of 40 characters in a text area. The same amount is used in this form.

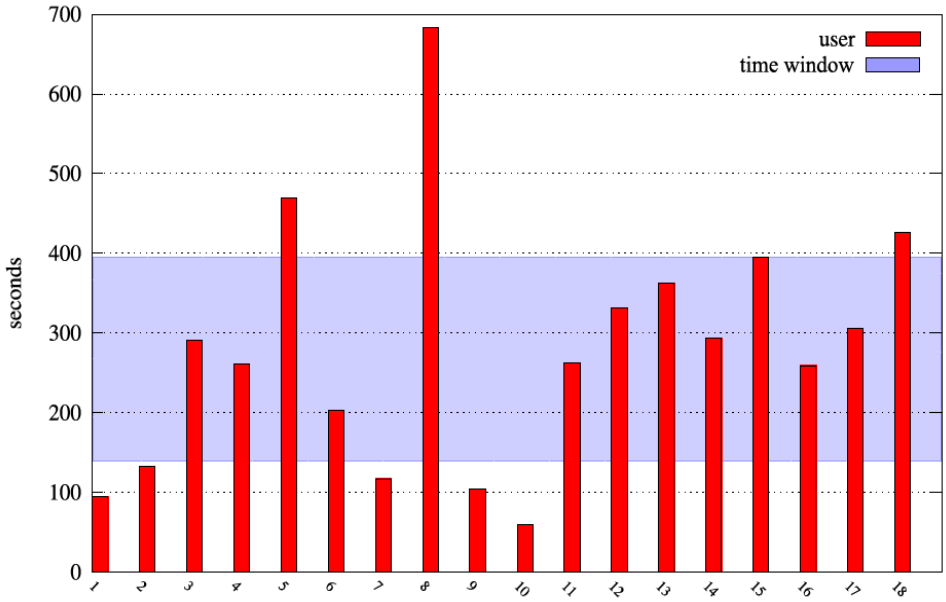
Two different sums are specified. The first is the sum of all required fields. This specifies the minimum time an average user should need to fill in the form. The second sum relates to all input fields. This sum specifies the maximum time.

Table 2 Time estimation

Element	*Size*	*Max*	*Required*	*Time calculation = total time*
Radio	10	-	Yes	10 ∗ 1.60 s = 16.00 s
Text	10	10	Yes	10 ∗ 2.12 s = 21.20 s
Text	10	10	Yes	10 ∗ 2.12 s = 21.20 s
Text	5	5	No	5 ∗ 2.12 s = 10.60 s
Text	40	60	Yes	20 ∗ 2.12 s = 42.40 s
Text	40	60	No	20 ∗ 2.12 s = 42.40 s
Text	40	60	No	20 ∗ 2.12 s = 42.40 s
Text	40	60	No	20 ∗ 2.12 s = 42.40 s
Text	40	60	Yes	20 ∗ 2.12 s = 42.40 s
Check	-	-	No	= 3.84 s
Check	-	-	No	= 3.84 s
Radio	3	-	No	3 ∗ 1.60 s = 4.80 s
Text area	60 × 5	-	No	40 ∗ 0.99 s = 39.60 s
Check	-	-	No	= 3.84 s
Check	-	-	No	= 3.84 s
Check	-	-	No	= 3.84 s
Check	-	-	No	= 3.84 s
Text	40	60	No	20 ∗ 2.12 s = 42.40 s
Check	-	-	Yes	= 3.84 s
			Sum required	= 147.04 s
			Sum all	= 394.68 s

To justify these sums, the web page was analysed for $3\frac{1}{2}$ month. During this period, 18 users filled in this form using a desktop browser. The dwell times of these users are presented in Figure 8. The vertical bar represents the time window between the minimum and maximum time. Only three users needed more than the specified time. One of them needed far too much time. Reasons for this might be that the form was too complicated, the user was disturbed by something, etc. The users who were faster might be expert users surfing the web. Expert users are faster making use of the browser. Unfortunately we did not ask the user about their experience. Altogether, the users dwell times fit very well in the time window.

Figure 8 Time users spent on form (see online version for colours)



5 RESTful web service

To retrieve all input elements on a web page, a RESTful web service has been developed. As input the web service receives the HTML content of every single web page. This HTML content is analysed and a JavaScript Object Notation (JSON) structure is generated. According to Richardson and Ruby (2007), our web service represents an algorithm. An algorithm has to use the HTTP method *POST* and receives the data, in our case the HTML content of one web page, in the *POST body* of the HTTP request. Parameters have to be specified in the URI. In our case we do not need any parameters. The result is included in the *POST body* of the HTTP answer. In our case it is a JSON structure.

Now the functionality of the web service is presented. The service uses the PHP Simple HTML DOM Parser library (Chen, 2013) to parse the HTML content of a web page submitted in the *POST-body*. The library offers a method to find elements of a specified type. Using this method all elements for any input (`<input>`, `<select>`, and `<textarea>`) are iterated. During this process all interesting attributes are recorded. The result is a JSON structure containing all elements for any input including their attributes.

A JSON extract of the web page analysed in Section 4 is shown in Figure 9. The first two input elements represent single line text elements. The attributes `size` and `maxlength` are particularly interesting. The others represent two check boxes.

Figure 9 Extract of the JSON answer

```
{
   "text":[
      {"id":"startdate","type":"text","name":"start",
      "size":"10","maxlength":"10","value":""},
      {"id":"enddate","type":"text","name":"end",
      "size":"10","maxlength":"10","value":""},
      ... ],
   "checkbox":[
      {"id":"flowers","type":"checkbox","name":"flowers",
      "value":"flowers"},
      {"id":"heart","type":"checkbox","name":"heart",
      "value":"heart"},
      ... ]
   ...
}
```

The reason for using a RESTful web service is that the service can be used as a plugin in a crawler application. The crawler traverses a whole website and uses the web service to get specific information on every single web page. One part of the information are the containing input elements. More can be found in the future work in Section 7.

6 Related work

Similar calculations are done by Oyewole and Haight (2011). They analyse users trying to access or navigate a website. Their goal is to develop a guidance technique to assist users. The path users take within a website is calculated using the GOMS model (goals, operations, methods, selection), resp. the Card, Morgan, Newell – GOMS (CMN-GOMS) (Card et al., 1986). Tasks done by users are stripped down into simple activities. These activities are investigated and required times are assigned. Oyewole and Haight uses the times specified in Table 3.

It is interesting to look at the form used in our case study (see Section 4 using the GOMS model. Table 4 shows the calculation related to Table 3. Every action needs at least one mental act. The radio buttons need more mental acts. The reason are the different descriptions assigned to the buttons. A motion of the hand and a click with the mouse button is also necessary. In contrast to our model the GOMS model does not differentiate between short and long texts. In general, all calculations show higher values. Only a check box is about twice as fast. Related to our form the time needed for all required fields is ca. 9.3% higher than in our model. Both results are similar.

Table 3 GOMS activities and times

Activity (short cat)	*Time*
Keystroke (K)	0.28 s
Type a sequence of *n* characters on a keyboard (T)	$n * 0.28$ s
Point with mouse to a target on the display (P)	1.1 s
Press or release mouse button (B)	0.1 s
Click mouse button (BB)	0.2 s
Home hands to keyboard or mouse (H)	0.4 s
Mental act of routine thinking or perception (M)	1.35 s
Waiting for the system to respond time (R)	0.1 s

Table 4 Time estimation with GOMS model

Element	*Size*	*Max*	*Required*	*Time calculation = total time*
Radio	10	-	Yes	$10 * M + H + BB = 14.10$ s
Text	10	10	Yes	$M + H + BB + 10 * T = 24.75$ s
Text	10	10	Yes	$M + H + BB + 10 * T = 24.75$ s
Text	5	5	No	$M + H + BB + 5 * T = 13.35$ s
Text	40	60	Yes	$M + H + BB + 20 * T = 47.55$ s
Text	40	60	No	$M + H + BB + 20 * T = 47.55$ s
Text	40	60	No	$M + H + BB + 20 * T = 47.55$ s
Text	40	60	No	$M + H + BB + 20 * T = 47.55$ s
Text	40	60	Yes	$M + H + BB + 20 * T = 47.55$ s
Check	-	-	No	$M + H + BB = 1.95$ s
Check	-	-	No	$M + H + BB = 1.95$ s
Radio	3	-	No	$3 * M + H + BB = 4.65$ s
Text area	60×5	-	No	$M + H + BB + 40 * T = 93.15$ s
Check	-	-	No	$M + H + BB = 1.95$ s
Check	-	-	No	$M + H + BB = 1.95$ s
Check	-	-	No	$M + H + BB = 1.95$ s
Check	-	-	No	$M + H + BB = 1.95$ s
Text	40	60	No	$M + H + BB + 20 * T = 47.55$ s
Check	-	-	Yes	$M + H + BB = 1.95$ s
			Sum required	= 160.65 s
			Sum all	= 471.75 s

Now we present related studies which deal with web usability and web statistics in general.

Atterer et al. (2006) built a system to track all user interactions with a browser page including mouse pointer, text input, and page scrolling. Similar to our approach, the interaction is tracked by a JavaScript but the data is transmitted using an image object. The web pages are modified automatically by a proxy server to enable the tracking. The

proxy server also protocols the usage data. The mouse-over event is recorded for every HTML element in the same way as in our study.

The tool Wusab is presented in Atterer (2008). Atter claims that automated usability validation is one of Wusab's features. During the website development the validation runs at regular intervals. In our tool the usability is checked on the request of the usability engineer. The Wusab tool downloads and analyses HTML code. The analysis is done using guidelines such as *images should have alt text*. Our tool also analyses the web page but offers more complex criteria such as the input elements used, readability (Martin, 2010), or subject (Martin, 2012).

Another tool is Google Analytics (2013) which is very powerful analysis tool. It works with its own database, whereby statistics can be provided independently of the web server used. The integration into a web application is done using JavaScript. The tool collects various items of information and sends them to the database by loading an invisible picture. Whereas Google Analytics provides many different statistics concerning users, sources, contents and goals, no information, however, is available on the visiting paths of the users. Furthermore, Google Analytics does not analyse the website itself.

7 Summary and future work

In this paper, we presented a method to estimate time requirements related to web forms. The goal was to find out the amount of time a normal user needs to fill in a form on a web page. First we presented a study. We examined how long it takes test persons to fill in single input elements of a form on a web page. The times spent by 80 persons to fill in the input fields were discussed in detail. As a result, Table 1 has been created. In Section 6 we demonstrated that our results are similar to these of the GOMS model. We believe that our model is more easy to use. In the study 80 persons were observed. To address a special group of users, e.g., older people, the study must be expanded. Right now we do not have enough persons to generate conclusions on special groups. Apart from that the results are fine.

To demonstrate the estimation of the average time needed by one user to fill in any web form a case study was presented. A form of a small website was analysed and a time window was calculated. This time window was evaluated in respect to real usage data.

In addition, a RESTful web service to automate the analysis of a web page was presented. The web service records all input fields. This web service supplements a set of services which analyse, e.g., the readability or the main subject of a web page. All these web services are used by a crawler which allows to automatically traverse a complete website. The result are measurements (see Martin, 2010) to classify the content of a website.

In Martin (2009), we presented a tool which can be used to analyse how people use a website. This tool is able to recognise paths that a single user or groups of users have taken. Additionally, the tool determines the corresponding dwell times. These dwell times can be validated by calculating the times described above for the forms offered.

Our next aim is to figure out whether it is possible to determine usability problems automatically. The measurements to classify the content of a website and the real usage data have to be investigated. Using semantic web technologies it should be possible to locate usability problems within a website.

References

Atterer, R. (2008) 'Model-based automatic usability validation – a tool concept for improving web-based UIs', in *Proceedings of the 5th Nordic Conference on Human-computer Interaction: Building Bridges*, pp.13–22, ACM, New York, NY, USA.

Atterer, R., Wnuk, M. and Schmidt, A. (2006) 'Knowing the user's every move – user activity tracking for website usability evaluation and implicit interaction', in *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, pp.203–212, ACM, New York, NY, USA.

Card, S.K., Moran, T.P. and Newell, A. (1986) *The Psychology of Human-Computer Interaction*, Lawrence Erlbaum Associates, Hillsdale, NJ, USA.

Chen, S.C. (2013) *PHP Simple HTML DOM Parser*, Technical report [online] http://simplehtmldom.sourceforge.net/ (accessed 23 October 2013).

Google (2013) *Google Analytics* [online] http://www.google.com/analytics/ (accessed 23 October 2013).

Martin, L. (2009) 'A tool to estimate usability of Web 2.0 applications', in *11th IEEE International Symposium on Web Site Evolution*, September.

Martin, L. (2010) 'Gathering information about Web 2.0 applications for usability engineering', in *Seventh International Conference on the Quality of Information and Communications Technology*, pp.482–486, September.

Martin, L. (2012) 'Subject classification of web pages', in *IADIS International Conference WWW/Internet*, pp.298–306, October.

Oyewole, S.A. and Haight, J.M. (2011) 'Determination of optimal paths to task goals using expert system based on Goms model', *Comput. Hum. Behav.*, March, Vol. 27, No. 2, pp.823–833.

Richardson, L. and Ruby, S. (2007) *RESTful Web Services*, O'Reilly, Sebastopol, CA, USA.