Framework for Relative Web Usability Evaluation on Usability Features in MDD

Shinpei Ogata(✉), Yugo Goto, and Kozo Okano

Shinshu University, 4-17-1, Wakasato, Nagano 380-8553, Japan
{ogata,okano}@cs.shinshu-u.ac.jp, 12t5033c@shinshu-u.ac.jp

Abstract. Web usability in business applications is crucial for enhancing productivity and preventing critical user errors. One of the effective methods to enhance the usability is employment of usability features such as auto-complete and input validation. Then, such employment should be designed so as to conform to actual end-users. However, its evaluation forces developers to expend a lot of effort to design an application, to create its Web prototypes, to observe user operations with the prototypes, and to assess the usability of the employed usability features. In this paper, a framework is proposed for evaluating the usability depending on the employment efficiently even if the developers are non-usability-specialists. Our framework has characteristics as follows so that the developers can easily determine the usability with low creation costs. (1) Support for creating Web prototypes, observing user operations, and assessing the usability are integrated centered on a model-driven approach. (2) The usability can be evaluated relatively and quantitatively by recording user operations and analyzing the resulting logs.

Keywords: Model driven development · Screen transition model · Usability evaluation · Business web application · Operability · User error protection

1 Introduction

Web usability in business applications is crucial for enhancing productivity and preventing critical user errors. Efficiency and the ability to avoid user errors belongs to "Operability" and "User error protection" respectively as sub-characteristics of "Usability" [1].

One of the effective methods to enhance the usability is employment of usability features [3–5] such as auto-complete and input validation. Developers should evaluate the effectiveness of such employment in the early steps of application development because failure of the employment often causes fundamental restructuring a radical modification of the design of the interaction between the users and the application.

C. Bogdan et al. (Eds.): HCSE 2016/HESSD 2016, LNCS 9856, pp. 73–85, 2016.
DOI: 10.1007/978-3-319-44902-9_6

However, the evaluation forces developers to expend a lot of effort to design an application, to create its Web prototypes, to observe user operations with the prototypes, and to assess the usability of the employed usability features. What is worse, such evaluation is often iterated many times to adjust the employment. In addition to the effort problem, the observation and assessment requires a high degree of special knowledge even though usability specialists do not often participate in the development.

In this paper, a framework to improve the effort and special knowledge problems is proposed focusing on an important but limited area in the operability and user error protection. Our framework helps developers to efficiently evaluate the usability of usability features employed with low creation costs even if the developers are non-usability-specialists. The evaluation process is supported by our framework as follows.

Web prototype creation: Several different Web prototypes employing usability features can be generated from a screen transition model by applying our previous work [7] in order to mitigate the effort problem.

User operation observation: User operation logs consisting of browser events and input values are recorded in order to assess the operability and user error protection with input time and user errors respectively.

User operation assessment: Many raw logs are processed in order to compare them relatively and quantitatively. This process also aims to help the developers to grasp the result of the observation easily.

The effectiveness of our framework is preliminarily evaluated by applying it to a part of a virtual e-commerce application. As a result of the evaluation, our framework quantitatively showed how the usability features affected the usability.

2 Usability in Ordinal Use of Business Web Applications

Usability generally covers various aspects of appropriateness recognizability, learnability, operability, user error protection, user interface aesthetics and accessibility according to ISO/IEC 25010 [1]. Enhancing usability can also enhance productivity of companies [2].

2.1 Scope Limitation

Our research aims to help skilled end-users to operate their Web application efficiently and correctly in normal use. In addition, our research also assumes that such end-users can be easily specified by a development project. Under this condition, the scope of usability is limited as follows.

– The priority of the appropriateness recognizability, learnability, user interface aesthetics, and accessibility is low in comparison with the operability, and user error protection because we target “skilled and specified” users.

– Each skilled office worker knows generally an important or laborious part of his/her work even if he/she has never operated any existing applications. This premise is advantageous in planning usability evaluation as follows.

1. Usage scenarios can be prepared concretely with hearing requirements for the actual users. For "unspecified" users, such preparation is difficult. Here, a usage scenario means concrete input/output values for achievement.
2. Although a skilled office worker often makes careless mistakes such as incorrect inputs, he/she does not make serious mistakes. Consequently, the evaluation does not need excessive coverage of user operation paths extended over irrelevant use cases.

These premises are important to make the iterative usability evaluation more realistic because the number of combinations of usability features is too numerous to evaluate the usability together with users.

2.2 Usability Measurement

Fundamental metrics, which form an essential guideline for improving the operability and user error protection, are defined as follows so that developers who are non-usability-specialists can easily understand effective usability feature employment.

Input time. Input time is directly related to not only the operability but also the productivity. In our research, input time is measured variously as the time between specific browser events, as explained in Sect. 3.4.

User error. Preventing user errors makes the productivity more reliable. In our research, user errors are defined as follows.

– One is a sequence of user operations that differ from the expected sequence of user operations. The expected sequence means the sequence of user operations that developers, customers or actual users can accept as correct and non-redundant.
– The other error is the input of incorrect values. The expected input value is a value that developers, customers or end-users can accept as correct.

3 Proposed Framework

3.1 Overview of Proposed Framework

A lot of usability evaluation methods have been proposed [6]. Also, methods focusing on metrics which are measured from user operation or access logs have been proposed [8–10]. As far as we know, there is no method of systematically evaluating the usability feature employment in order to take one step further.

Figure 1 shows the overview of our framework. This framework consists of three phases. The first one is the "preparation" in which several different Web

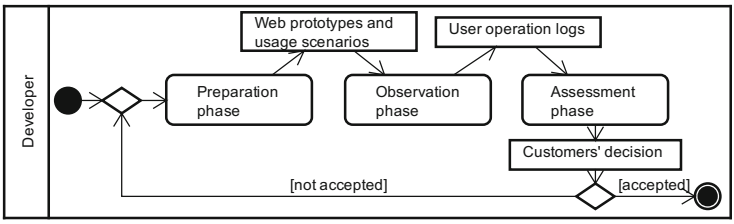


Fig. 1. The overview of our framework.

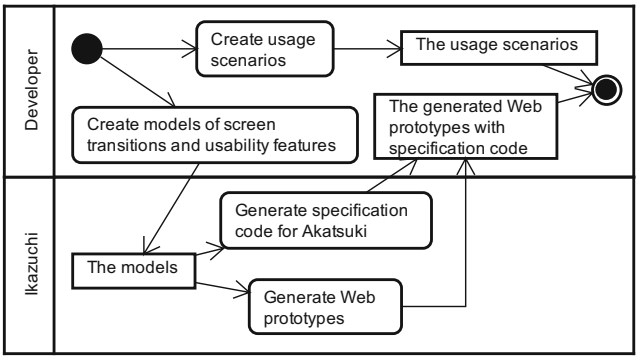


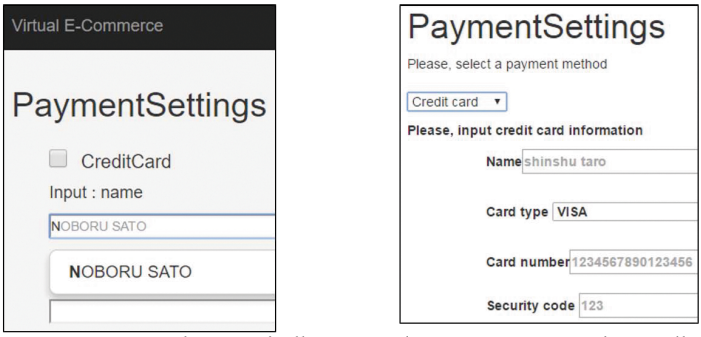
Fig. 2. The preparation phase of our framework.

prototypes and scenarios for evaluating the usability are created. The second one is the "observation" in which logs of user operations as browser events are recorded. The last one is the "assessment" in which the logs among different Web prototypes are analyzed and compared. Each of these processes is explained in detail from Sect. 3.2. Our framework provides three tools called Ikazuchi, Akatsuki, and Hibiki respectively, for the process support.

3.2 Preparation Phase

Figure 2 shows the preparation phase of our framework. Developers create Web prototypes and usage scenarios in this phase. Generally, absolute evaluation is difficult for developers who are non-usability-specialists because they may be not able to determine the usability from its evaluation result. Therefore, relative evaluation comparing user operation logs between different Web prototypes is supported in our framework. In the relative evaluation, the developers can easily understand which prototype is better for users. However, the cost for creating several different Web prototypes is not low. To mitigate this problem, our framework applies Ikazuchi [7] which is a Web prototype generation tool.

Ikazuchi. Ikazuchi is a tool for generating Web prototypes from a design model of screen transitions. This tool and model have been proposed in



(a) A prototype generated automatically (b) A prototype created manually

Fig. 3. Parts of prototypes to which specification code can be applied.

previous work [7]. A main characteristic of the method of the previous work [7] is separation of concerns between the usability features and screen transitions in this model. Then, different Web prototypes can be generated from this model. For example, a Web prototype employing the usability features can be generated by interpreting the whole model as it is. On the other hand, another Web prototype employing no usability features can be generated from the screen transition part without any usability features.

Thereby, Web prototypes that are compared in relative evaluation can be obtained at a low creation cost. Ikazuchi tentatively supports four usability features: auto-complete, auto-save, undo, and validator, which can be attached to input items without changing screen's structure. The screen transition model can represent screens, screen components, and the items that are categorized into three types as Input, Output, and Link. Figure 3(a) shows a part of a Web prototype generated with the Ikazuchi. Figure 3(b) shows a part of a Web prototype created by hand.

Specification Code. The specification code represents correspondence between HTML tag ids and the model. An example of code written in JSON format is shown as follows. The code consists of a page title, input items, and link items.

Specification code

```
{"title": "PaymentSettings",
  "inputs":
   {"name": ["input-977674685"],
    "cardType": ["input-836427078"],
    "cardNumber": ["input-1322642290"],
    "expirationYear": ["input-2121199924"],
    "expirationMonth": ["input-431570856"]},
  "links": {"next": ["link-520162288"] }
}
```

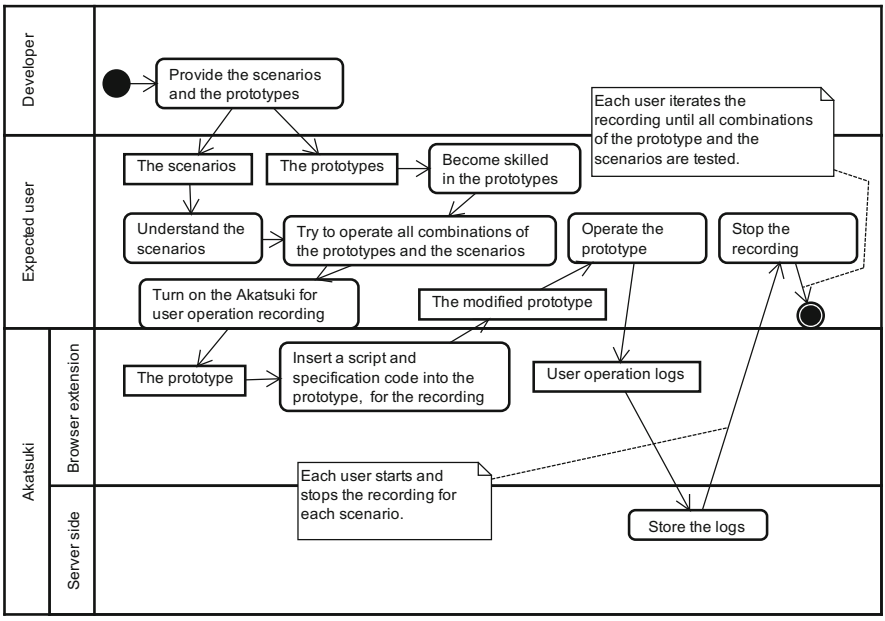


Fig. 4. The observation phase of our framework.

This specification code can be generated for the generated Web prototypes with the Ikazuchi but not for the product. That is, our framework requires the hard coded values in the specification code so as to conform to the scenarios. Such simplicity is useful for manually inserting the specification code into a Web prototype even if the prototype was created manually. For our framework, the Ikazuchi was extended so as to insert specification code into a generated prototype in order to record user operations corresponding to the model. Specification code can be manually inserted into Web prototype created manually.

3.3 Observation Phase

Figure 4 shows the observation phase of our framework. The precondition of this phase is preparation of both the scenarios and the Web prototypes containing the specification code. That is, developers do not always need the Web prototypes which can be generated with Ikazuchi but which are too simple. The following actions must be conducted by expected users before recording user operations.

- The users have to become skilled in each prototype because the focus is on operability and user error protection and not learnability. By excluding learnability evaluations, it's approximately valid to reuse the prepared scenarios between use of the different prototypes.
- The users have to understand and validate the prepared scenarios by using each prepared prototype because it's difficult to imagine that skilled office

Table 1. The kinds of user operations and recording timing by Akatsuki

User operation	Recording timing
Change of window size	(1) A `window.onResize` event occurred, or (2) a script for the recording was inserted
Change of scroll position	(1) A `window.onScroll` event occurred, or (2) the script was inserted
Change of mouse position	(1) A `document.onMouseMove` event occurred
Operation of mouse buttons	(1) A `document.onMouseDown` event occurred, or (2) a `document.onMouseUp` event occurred
Input of keys	(1) A `document.onKeyDown` event occurred, or (2) a `document.onKeyUp` event occurred
Start of item input	(1) A `focus` event on the item occurred
End of item input	(1) A `blur` event on the item occurred
Screen transition	(1) A `click` event on a link occurred

workers would misunderstand the scenarios even if they often make mistakes. The validity of normal flow can be ensured at least by the validation.

With the above assumption, the users operate Web prototypes in accordance with the scenarios. The Akatsuki tool is utilized to record the user operations.

Akatsuki. Akatsuki consists of a browser extension and a server side program for recording user operations as browser events. The extension inserts a logging script into a rendered Web page, while the server side program receives the user operations from the extension and stores them into a database. The extension is implemented as a Google chrome extension. The server side program and database are implemented with PHP and MySQL respectively.

Table 1 shows both the kinds of user operations recorded by Akatsuki and the timing of the recording. Table 2 shows the log contents for each kind of user operation. The page title and input, output, and link items are recorded together with their names represented in the screen transition model so that the developers can review the model on the basis of the analyzed logs. The specification code is used when an HTML tag as an event target is transformed into such names.

3.4 Assessment Phase

Figure 5 shows the assessment phase of our framework. In this phase, the developers relatively evaluate the usability feature employment of the prototypes on the basis of the user operation logs. The raw logs are numerous and difficult to understand. To mitigate this problem, our framework provides the Hibiki tool which analyzes the raw logs and transforms them into useful information.

Table 2. The user operations and corresponding log contents created by Akatsuki

User operation	Log contents
Commons	The recorded timestamp, page title, and recording id of a log
Change of window size	The width and height of a window
Change of scroll position	The x axis of the horizontal scroll of a page and the y axis of the vertical scroll
Change of mouse position	The x and y axis of the cursor in a browser
Operation of mouse buttons	The state (`press` or `release`) of a button and the button type such as `left`, `right`, `middle`, `undefined`
Input of keys	The state (`press` or `release`) of an alphabet or number key, the key code, and the state (`press`) of specific keys such as Shift, Ctrl, Alt
Start of item input	The name of an input item when the item was focused
End of item input	The name of an input item and the value of the item when the item was focused
Screen transition	The name of a link when the link is clicked

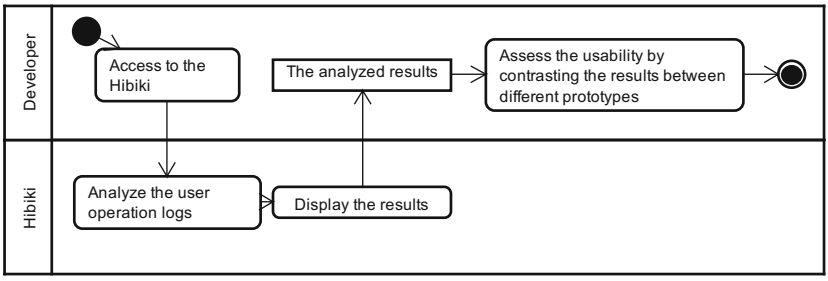


Fig. 5. The assessment phase of our framework.

Hibiki. Hibiki is a server side program in PHP that summarizes the recorded logs into user operation time and input values. The largest unit of the summarized information shows the user operations from the start of the recording to the end. The information in this unit shows the result when a certain user operated a certain prototype in accordance with a certain scenario. Table 3 shows the summarized information shown by Hibiki. Various differences can be analyzed by collecting this information. For example, differences in the users, in the prototypes, in the scenarios, and in combinations of them can be shown.

4 Preliminary Evaluation

4.1 Overview

This evaluation aims to determine whether the difference in the usability resulting from usability feature employment can be quantitatively recognized from the data shown by Hibiki. A part of a virtual e-commerce application is adopted as an application example because a partial prototype is often used to evaluate specifications in early stages of development.

For evaluating the potential effectiveness of the proposed method for observing user operations under conditions assuming actual development is important, weakly restricted prototypes and scenarios which are often create by actual developers are used. In this evaluation, we observed user operations with weakly

Table 3. The kinds of analyzed information by Hibiki

Information type	Explanation
Scenario achievement time [millisecond]	Time from the start of user operation recording to the end. The individual time is measured per scenario
Page sojourn time [millisecond]	Time between screen transitions. Both the individual and cumulative times are measured per page
Input time [millisecond]	Time focusing on any input item on a page. The individual time is measured per page
Mouse input time [millisecond]	Time from a button press to the release on a page, except for focus change. The individual time is measured per release event
Click frequency [number of times]	The number of button clicks on a page. One click is countered per release event of the mouse button
Key input time [millisecond]	Time from a key press to the release on a page. This individual time is measured per release event of the key
Key type frequency [number of times]	The number of key types on a page. One key type is countered per release event of a key
Backspace key type frequency [number of times]	The number of backspace key types on a page. One key type is counted per release event of the key
Time interval to input next item [millisecond]	Blur time until focusing the next input item. The individual time is measured per focus event of the input item
Sequence of screen transitions	Sequence of displayed pages in chronological order
Sequence of input item focuses	Sequence of focused input items in chronological order
Values of input item	The last value of an input item in a scenario

restricted prototypes because such prototypes are similar to one obtained in an actual development process.

In this evaluation, there were two prototypes, six scenarios, and five users. In this paper, we call the two prototypes A and B. A was automatically generated and B was created manually as shown in Fig. 3. Each prototype represents screen transitions from the `CartView` screen to the `PurchaseComplete` screen. This structure simply consists of five screens and transitions between them without any branches. The scale of each of the six scenarios was categorized as large, middle or small resulting in two of each category. Combinations of the prototypes, scenarios, and users were determined so that no bias in learning resulted.

4.2 Result

From the evaluation, the total number of the logs obtained was 176,278. Of the 30 combinations that we tried, 26 of them resulted in valid data.

Table 4 shows the result of comparing the average input times (avg.) between A and B where the difference between the averages is three or more seconds. ID 1, 2, 3, 4, and 6 shows that the users were able to efficiently operate A than B, in average. All of these items of A employed an auto-complete as a usability feature although these items of B employed no usability features. On the other hand, all items of ID 5, 7, and 8 of B made the user operation more efficient. The items of ID 7 and 8 of B employed a spinner with a default value as "1" although the items of A employed no usability features. Regarding the item of

Table 4. The result for operability evaluation

ID	Input item	Avg. of A [sec.]	Avg. of B [sec.]
1	PaymentSettings .name[0]	7.993	13.508
2	PaymentSettings .cardNumber[0]	8.463	15.442
3	BillingAddressSettings .billingAddress[0] .address[0]	9.051	24.116
4	BillingAddressSettings .billingAddress[0] .phoneNumber[0]	10.906	14.551
5	ShippingAddressSettings .shippingAddress[0] .address[0]	26.404	21.995
6	ShippingAddressSettings .shippingAddress[0] .assignedItems[0] .name[0]	7.822	12.533
7	ShippingAddressSettings .shippingAddress[0] .assignedItems[0] .quantity[0]	5.299	1.774
8	ShippingAddressSettings .shippingAddress[0] .assignedItems[1] .quantity[0]	4.376	0.183

Table 5. The result for user error protection evaluation

Kinds of the errors	Freq.	Example
[MI] Mis-input a 1-byte character instead of a 2-byte one	11	
[MU] Misunderstanding of the scenarios	11	"Kawakami" as a last name was inputted instead of "Kami kawa".
[O] Omission of a part of a string	5	"5300001" as postal code was inputted instead of "530-0001"
[T] Typo	2	"Satoru" as a first name was inputted instead of "Satoshi". Both of the "ru" and "shi" are one 2-byte characters in Japanese
[MC] Mis-capitalization	1	"noboru" as the name on a credit card was inputted instead of "NOBORU"

ID 5, both of A and B however employed the same input method without any usability features.

As for user error protection evaluation, we analyzed input mistakes against 222 values expected in the total of all scenarios. A few of the expected values were excluded because some values that were inputted through a date picker written in Javascript were not collected. A total of 906 values were obtained for the 222 expected values. 30 of the actual values were different than the expected ones.

Table 5 shows user errors discovered in the evaluation. Usability features to decrease those errors were not employed in both A and B. The errors such as MI, MC, and a part of O can be automatically corrected employing functions for formatting characters. A part of the errors of MU, O, and T may be avoidable by employing auto-completes because the users inputted the correct value at least once in a scenario although he/she inputted the wrong value in another scenario. Meanwhile, auto-completes were useful to avoid most of these errors but it promoted the error a little. For instance, one of the users inputted the number of a credit card with blind acceptance of the auto-complete even though the completed value was different than the expected value.

4.3 Discussion

The effectiveness of usability feature employment was shown quantitatively by utilizing the information analyzed by Hibiki. Hibiki made the analysis of the preliminary evaluation result efficient even though there were numerous logs. However, it seems that the result shown as ID 5 of Table 4 did not depend on the usability features. Consequently, various aspects such as user characteristics and qualitative aspects should be considered for more precise evaluation. Tool support is also needed to make the usability evaluation practical.

In the preliminary evaluation, there were still various non-stable factors such as differences in scenario properties such as the amount of input/output data, rigorous user operation sequences, and exhaustiveness and differences in prototype properties such as layout and input methods. These factors may make the effectiveness of usability feature employment ambiguous. Therefore, a more rigorous evaluation process well supported by proper tools should be considered.

In addition, Ikazuchi can handle a few number of usability features yet. The strategies of attaching usability features, the combinations of usability features and the strategies of comparing the obtained logs should be considered for observing a complex combinations of usability features by improving the capability of Ikazuchi. We plan to extend Ikazuchi to support more existing usability features [3–5].

5 Conclusion

A framework for evaluating the operability and user error protection relatively and quantitatively was proposed in this paper. In our framework, Web prototype creation, user operation observation, and usability assessment are supported so as to obtain the prototypes, the user operation logs, and the summarized logs semi-automatically. Our framework strictly controls the evaluation conditions to clarify the effectiveness of usability feature employment. As a result of the preliminary evaluation, this paper quantitatively showed that the usability changed depending on usability feature employment.

As future work, we plan to extend Ikazuchi so as to deal with more usability features and their employment algorithms. We also attempt to extend Hibiki so as to more easily grasp differences between different employment methods by applying artificial intelligent algorithms. In addition, we consider how to obtain the exhaustive scenarios by applying a model-based testing approach. Finally, a large scale and rigorous evaluation applying experimental research approaches is planned to show more reliably the effectiveness of our framework and to evaluate the user error protection from the aspect of user mis-operation sequences.

Acknowledgment. This work was supported by JSPS KAKENHI Grant Number JP15K15972.

References

1. ISO/IEC: systems and software engineering - systems and software quality requirements and evaluation (SQuaRE) - system and software quality models. ISO/IEC Std. 25010: 2011 (2011)
2. Nielsen, J.: Usability Engineering. Morgan Kaufmann Publishers Inc., San Francisco (1993)
3. Folmer, E., Gurp, J.V., Bosch, J.: A framework for capturing the relationship between usability and software. Softw. Process Improv. Pract. **8**(2), 67–87 (2003)
4. Juristo, N., Moreno, A.M., Sanchez-Segura, M.-I.: Guidelines for eliciting usability functionalities. IEEE Trans. Softw. Eng. **33**(11), 744–758 (2007)

5. Roder, H.: Specifying usability features with patterns and templates. In: First International Workshop on Usability and Accessibility Focused Requirements Engineering, pp. 6–11 (2012)
6. Insfran, E., Fernandez, A.: A systematic review of usability evaluation in web development. In: Hartmann, S., Zhou, X., Kirchberg, M. (eds.) WISE 2008. LNCS, vol. 5176, pp. 81–91. Springer, Heidelberg (2008)
7. Kamimori, S., Ogata, S., Kaijiri, K.: Automatic method of generating a Web prototype employing live interactive widget to validate functional usability requirements. In: 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence, pp. 9–14 (2015)
8. Atterer, R., Wnuk, M., Schmidt, A.: Knowing the user's every move: user activity tracking for website usability evaluation and implicit interaction. In: 15th International Conference on World Wide Web, pp. 203–212 (2006)
9. Nakamichi, N., Sakai, M., Shima, K., Matsumoto, K.: Detecting low usability web pages using quantitative data of users' behavior. In: 28th International Conference on Software Engineering, pp. 569–576 (2006)
10. Pansanato, L.T.E., Rivolli, A., Pereira, D.F.: An evaluation with web developers of capturing user interaction with rich internet applications for usability evaluation. Int. J. Comput. Sci. Appl. **4**(2), 51–60 (2015)