Automating the Evaluation of Usability Remotely for Web Applications via a Model-Based Approach

Nouzha Harrati [1,2], Imed Bouchrika [2], Abdelkamel Tari [1] and Ammar Ladjailia [2,3]
[1] Department of Computer Science, University of Bejaia, Algeria
[2] Faculty of Science and Technology, University of Souk Ahras, Algeria
[3] Department of Computer Science, University of Annaba, Algeria
n.harrati@univ-soukahras.dz

***Abstract*—Usability for software systems has emerged as an integral part of the continuous commercial success of IT companies. This is partly due to the vital need to satisfy customers' goals for systems becoming pervasive and ubiquitous within our daily life. In this research study, we have explored the use of task models to define how the user should interact with a given system. Based on empirical data collected from end-users participating within the usability evaluation of a web application, data analysis is conducted to infer the usability degree. This is carried out in compliance with the defined task model and usability metrics describing efficiency of use. The proposed approach is a milestone towards automating usability evaluation as most of the studies are reporting manual-based methods to assess the usability of software systems. Experimental results performed to assess the usability of a website shows the potency of the system to discover usability setbacks that can be addressed to improve the user experience.**

***Index Terms*—remote testing, task modeling, usability evaluation.**

I. Introduction

Positive user experience is of prime importance for software development playing vital role for the continuous commercial success of software companies. In fact, the increase of customers base and loyalty are totally related to the better design of products. Usability of software products is a key characteristic to achieve the acceptance of users regardless of their background, experience or orientation. Usability is defined as the extent to which a product can be easily used by specified users to achieve certain goals with effectiveness, efficiency and satisfaction. In practice, the usability aspect of software products is marginalized during the classical stages of software development life-cycles pushing and devoting more efforts resources into the software back-end to address the functional requirements. In fact, regardless of how software are neatly coded or sophisticated, recent studies of software sales reports that software failures are due to usability reasons where simply the user does not know how to use the purchased product [1]. It is no doubt that usability is now recognized as an important software quality attribute, earning its place among more traditional attributes such as performance, robustness and security.

The process of usability evaluation (UE) consists of methodologies for measuring the ease-of-use aspects of the user interface for a given software system and identifying specific problems. In fact, Usability evaluation plays a vital role within the overall user interface design process which undergoes continuous and iterative cycles of design, prototyping and testing. Evaluating the usability of interactive systems is itself a process involving various activities depending on the method utilized [2]. Empirical-based usability methods require the participation of end users who are instructed to interact with the software system. Meanwhile, their behavior and interaction with the system are recorded and observed by an expert. results are obtained from the users through interviews and questionnaires where they are asked for their opinions and concerns in addition to possible suggestions of how to better improve the interface design and its usability. In fact, one of the challenges in software development is to involve end users in the design and development stages so as to observe and analyze their behavior in order to collect feedback in effective and efficient manner. Alternatively, usability evaluation can be carried out through inspection methods which aim to identifying interaction problems within the interface or a prototype [12] without the involvement of end users. The interface is assessed by an expert or usability consultant for compliance to a set of predefined usability guidelines or conventional set of heuristics [3].

Because of the dearth of approaches devoted to the automated evaluation of the usability aspect for web applications, we explore in this research study the use of a task descriptor to define how the user should interact with a given system. Based on empirical data collected from end-users participating within the usability evaluation of the system, data analysis is conducted to infer the usability level. This is carried out in compliance with the defined task model and usability metrics describing efficiency of use. Experimental results performed to inspect the usability of a website shows the potency of the system to discover usability issues that can be addressed to improve the user experience. The proposed approach is a milestone towards the automation of usability evaluation as most of the studies are based on manual methods to assess the usability of software systems.

This paper is organized as follows. The next section outlines the previous approaches for automated usability evaluation of software systems. The theoretical description of the presented approach is described in sections 3. Section 4 is

devoted to show the experimental results attained for the usability evaluation using the proposed approach on a real case scenario.

II. Related work

Usability evaluation of web applications has received considerable attention since the advent of the web. This is partly due to the vital need to satisfy customer's goals for systems becoming pervasive within our daily life. Ivory and Hearst [2] presented a survey of tools for usability evaluation according to a taxonomy based on four dimensions: method class, method type, automation type and effort level. Ivory argued that the automation of usability evaluation would help to increase the coverage of testing as well as reduce significantly the costs and time for the evaluation process. Fernandez [3] surveyed the recent studies related to usability evaluation where they have categorized the different methods into broadly two main classes; empirical and inspection methods. However, the majority of the surveyed research studies are purely based on the manual or statistical analysis of recorded activity data for the participants. Automated testing of interactive systems enables the possibility of remote evaluation. Tullis [4] conducted a comparative experiment between remote and laboratory-based testing where they emphasized the advantages of remote evaluation in terms of costs and effectiveness.

Paganelli [5] worked on developing a desktop-based application for recording and analysing interaction logs for website systems based on a predefined task model. The activities to be performed on a website are specified using the notations for the ConcurTaskTrees environment [6] which provides a graphical representation for the hierarchical logical structure of the task model. Tiedtke [7] described a framework implemented in Java and XML for automated usability evaluation of interactive websites combining different techniques for data-gathering and analysis. Their system uses a task-based approach and incorporates usability issues. Atterer [8] presented an implementation of UsaProxy which is an application that provides website usage tracking functionality using an HTTP proxy approach. Recently Vasconceols [9] implemented an automated system called USABILICS for remote evaluation based on interface model. Tasks to be performed by a user are predefined using an intuitive approach that can be applied for larger web systems. The evaluation is based on matching a usage pattern performed by the user against the one conducted by an expert of the system providing a usability index for the probed application. Muhi [10] proposed a general framework for usability evaluation that can be tested in production systems. The framework takes as input an XML configuration file describing the positioning of the different interface elements of an application whilst user activities are logged into a separate XML file. A validator module is deployed to check the log-files according to semantic rules that are defined within the usability data model. Andrica *et al.* [11] presented the WaRR which is an automated tool that records and replays with high fidelity the interaction between users and modern web applications in this tool the recording functionality is embedded in the web browser, it has direct access to user keystrokes and clicks.

There are a number of commercially available tools that are used for recording user traces for usability purposes. CrazyEgg logs mouse events with the ability to visualize activity maps of the more popular locations of clicks on a page. Web Criteria Site Profile is another tool used to assess simple attributes of usability including page loading time and ease of finding content. This is based on automated agents browsing the website to retrieve data making use of the GOM model. Web TANGO is a software that employs the Monte Carlo simulation and information retrieval methods to predict the user's behavior and navigation paths. This is based on data acquired from extensive experiments conducted against websites nominated as successful having received higher user ratings.

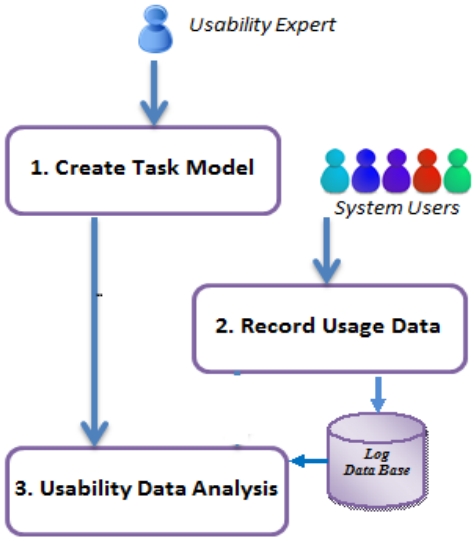

Fig.1. Proposed Framework for Remote Usability Evaluation

III. Proposed Approach

The acceptability of interactive systems is usually based on their utility and usability. The utility refers that the application offers a service or a functionality for the user to achieve some goal. Meanwhile the usability factor concerns how easy and efficient the task is performed to achieve such utility. In order to assess the usability aspect of a given web application, the proposed system consists of three main phases: i) Task Modeling ii) Usage Tracking, iii) Data Analysis. An overview describing the proposed approach is shown in Figure (1). During the first stage, a task model is laid out to describe how to interact with the system. The modeling which is based on a newly proposed tree-based graphical notation, is usually performed by a usability expert. In the following phase, usage data is tracked and recorded from users who are usually invited to test the system remotely. Finally, automated analysis of the collected usage data is carried out to assess how users data adheres and complies well with the defined task model. Based on usability metrics, the system can be trained to infer how usable the system is.

A. Task-Based Descriptor

Traditionally, usability is measured through monitoring the completion of certain goals or a task provided by an interactive system. The satisfaction level can be evaluated by a usability expert monitoring user's activities or through asking users to fill in questionnaires. However, automated verification and usability evaluation for achieving a defined set of tasks are proven to be a difficult task especially for web applications. This is partly due to the complex nature of web systems involving many interaction styles that can vary with different display hardware in addition to a large number of UI components and events rendering formal modeling of user behavior a challenging process. In this study, a fully automated system for formalizing user interaction with a given system guided through a set of rules describing certain goals to be achieved by the end user. This is done through defining a \textit{task model} by an expert to describe how the user should interact with the system. The task model is mainly utilized to capture all the interactions to be carried out by user. The compliance with the defined model by users infers that the system model believed to be the optimal use set by a usability expert matches the user model. This can reflect better usability. There are several approaches and notations for defining a task model for usability evaluation such as ConcurTaskTree (CTT) [6], Goals Operator Method and Selection rules (GOMS) [13] and Hierarchical Task Analysis (HTA) [14].

The tree-based graphical representation introduced by Harms *et al.* [15] for creating a task model from collected usage traces of users is being adopted as the basis throughout this research. In the same way as the CTT notation, task models should offer the designers only with high level details in order to focus on the overall interaction and flow of a user interface without becoming distracted by the low-level details by which the user interface is presented on various platforms and styles of interaction. In this research study, we propose a tree-based graphical representation for defining a task model that should describe the tasks, actions and goals to be performed by the user. The resulting task model tree represents all interactions a user can perform on given software interface. Tasks can be combined to describe higher level tasks. Using the tree-based visual notation, the task model is an ordered hierarchy of tasks or other elements to be performed in order to satisfy a specific goal for a task. In order to enable automation at later stages, goals for actions should have a way to infer automatically whether a task is completed successfully based on conditions and events. Consideration is made towards the expressiveness of the visual notation which defines the capability of the model to express user activities [16]. The proposed task modeler is implemented as online application for usability experts to create a task model for evaluating the designed interface of their software systems.

A task consists of actions to be performed to achieve a specific goal. This can be a basic task consisting mainly of simple actions such as clicking a submit button, page scrolling or typing a text into a text field. For each basic action, there should be a mapping to an event caused by performing the action. In addition, it can be a complex task composed from other subtasks and advanced control blocks such as filling a payment checkout form for an online shopping cart containing many widgets with a number of options and conditions to be verified. Various control blocks are employed for expressing the temporal relationship for task children which determines the number and order in which the subtasks must be performed by the user to achieve a goal. Control blocks include sequence, iteration and choice. The different notations used to describe visually the different modeling blocks are explained as follows:

- *Task* : refers to a complex or basic task to be performed by a user to achieve a goal. The syntax for creating a task is given as:

  Task : Goal Name
- *Sequence* : it describes an aggregated set of tasks that must be performed by the user through the specified order in which they appear.
- *NoOrder* : As opposed to the Sequence clause, this is used to define that the subtasks can be executed regardless of the specified order.
- *Iteration* : This refers to the case where the enclosed set of tasks must be executed by the user zero or more times depending on the specified cardinality.
- *Choice* : This is to specify that the user must choose a task among a list of given tasks.
- *Success* : This control block is employed to deduce that a task is completed successfully by the user. Criteria for inference include different checking conditions including simple event triggers to advanced verification as the use of regular expression matching. The syntax for using the *Success* clause is given as:

  Success : KeyUp : Enter

  To show that the parent task is performed successfully when the *enter key* is released after being pressed. Other event triggers include scroll, keyPress, mouseClick, mouseOver ... etc. The following example illustrates the use of regular expression using the *pattern* keyword to verify the validity of an email address:

  Success : pattern : [a-z0-9]+@[a-z]+.[a-z]{2,4}
- *Action* : This is the leaf of the hierarchical task tree referring to simple events.
- *ElementName* : It is used to specify the HTML component name that can be mapped to a given task where the *Success* can use to verify completion of the event. Other equivalent clauses can be used including *ElementID* or *ElementType*.

The same operators defined in the CTT are implemented within the proposed task modeling platform to add further flexibility and control for the defined task model. In the same way as the CTT tool, the same icons are added within the graphical notion. This includes as an example Task Enabling which refers that a task cannot be started until another task is completed successfully. For better expressiveness, the cardinality for a task can be specified indicating the possible repetition of a task using the conventional syntax using for UML modeling. For instance, *1..** refers to at least one or more. The placement of cardinality condition is done at the left side of the task box. As opposed to the work described by Long

[17] where an alike of programming language is presented for task modeling, procedures can be created within this study to encapsulate certain business logic. However, the main focus is devoted towards simplicity and usability hence avoid the necessity to re-invent a fully programming language that needs further training. Figure (2) shows an example for a task model creating a for login page.

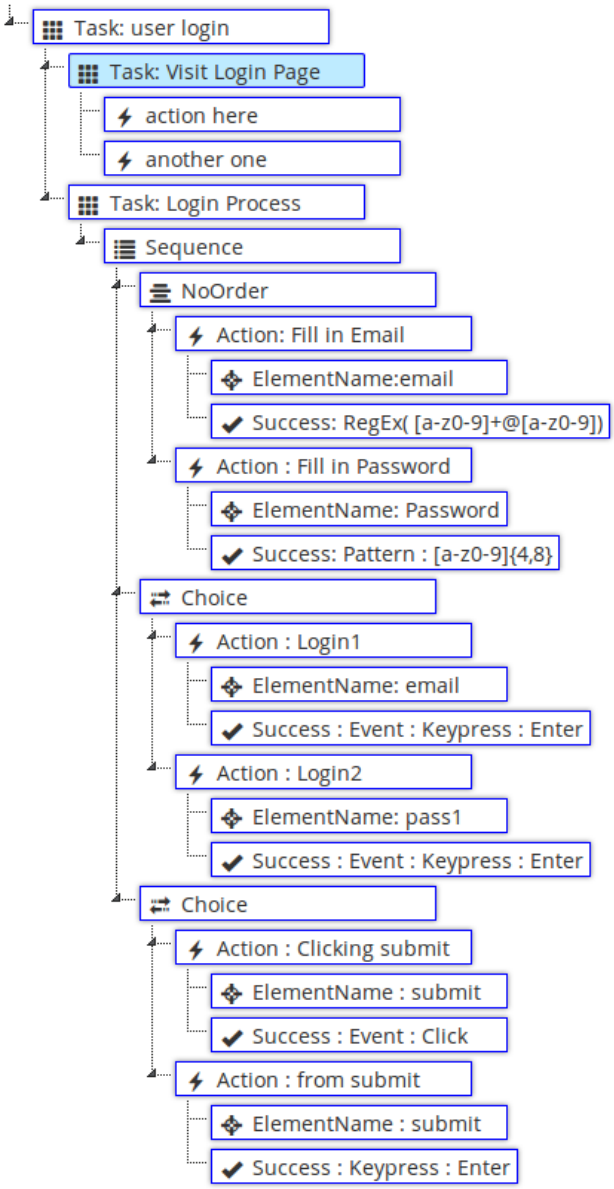

Fig.2. Tree-based Task Descriptor Example

The graphical diagram for the tree-based task model can be exported to XML format for portability, openness and interoperability reasons as an initial step for a transformation process to other models. Therefore, the XML file can be parsed to assist with for automation of the usability evaluation (UE). The Document Type Definition (DTD) of an XML document describing the structure or format is considered for its simplicity. Listing (1) shows the structure for the generated XML document of the proposed task descriptor.

B. Usability Data Collection

Because of the setback that computer applications have not been designed with an eye to user modeling [18], it becomes vital and crucial to gain access to the stream of user actions to get an insight for their experience. Consequently, various research studies and software tools were proposed to devise ways to extract and analyze useful usability information from user interfaces. For usability analysis, it is typical to automatically collect clicks, page views and visit duration in order to determine conversion rates and website traffic. For the course of this research, a JavaScript program is implemented to log all user activities performed when browsing a given website to get its usability assessed. To avoid the need to install third party software on the client machine such as Java virtual machine. This is one of merits of the approach to move towards unintrusive automated testing of web applications. The proposed tool is integrated by appending a single line of JavaScript code into the web page without the need for the website programmer to modify their existing application code. The appending can be done either from the server side or by using a custom browser plugin to automatically add the script code into the browsed website.

```
<!DOCTYPE Usability [
<!ELEMENT Usability (Name, Website, Task*) >
<!ELEMENT Name (#PCDATA) >
<!ELEMENT Website (#PCDATA) >
<!ELEMENT Task
(Sequence*,NoOrder*,Choice*,Iteration*,Action*))>
<!ATTLIST Task URL CDATA #IMPLIED >
<!ATTLIST Task Description CDATA #IMPLIED >
<!ELEMENT Sequence
      (Task*,NoOrder*,Choice*,Iteration*,Action* >
<!ELEMENT NoOrder
      (Task* Sequence*,Choice*,Iteration*,Action*>
<!ELEMENT Choice
     (Task*,Sequence*,NoOrder*,Iteration*,Action*>
<!ELEMENT Iteration
     (Task*, Sequence*,NoOrder*,Choice*,Action*>
<!ELEMENT Action Success>
<!ATTLIST Action name CDATA #REQUIRED >
<!ELEMENT Success Trigger,
(ElementName|ElementID|ElementType|URL)?,CondArg*)
<!ELEMENT Trigger (#PCDATA) >
<!ELEMENT ElementName (#PCDATA) >
<!ELEMENT ElementID (#PCDATA) >
<!ELEMENT ElementType (#PCDATA) >
<!ELEMENT Condition (#PCDATA) >
<!ELEMENT CondArg (#PCDATA) >
]>
```

Listing 1: DTD for the proposed task descriptor

Once the web page is loaded, the JavaScript tool is invoked registering event handlers which are called for all events of interest triggered by the user when interacting with the interface. The events include typing, cursor movement and mouse clicking. Recorded events should be always associated with the browser timestamps that describes the date and time information as timing is considered important for understanding the order of events performed by the user. For the events of typing and mouse clicking, identifying attributes for the HTML element of interest that triggered the event are recorded for every action to ease later matching between the

task descriptor and user data. These attributes include the node name, id, name and type. The type attribute is used to distinguish between form components such as radiobox, button and input text. Example of logged data for a mouse click is shown in Listing 2. The choice to record cursor movements is to measure the traveled distance of the mouse. For a continuous cursor motion, the starting and end points are recorded along with their timestamps in order to estimate the traveled Euclidean distance.

```
Event:{
  "Timestamps": "1436717509880",
  "Type": "MouseClick",
  "ElementID": "button2",
  "ElementName": "double click",
  "ElementNodeName": "input",
  "ElementType": "button",
  "CursorX": "200",
  "CursorY":"310",
  "URL": www.usability.ws
}
```

Listing 2: Example of logged data for a mouse click

The logged data is collected by the browser without interfering with any existing JavaScript code. Because the data is stored centrally on a remote server, the data is encoded in JSON format and transmitted back at regular intervals to the server for permanent storage. The submission into the server is based on a buffer of a particular size to avoid bandwidth bottle neck and network problems. A client session key is created for every user with an expiry time of ten minutes. This is used to map received data to their respective user. The IP address is also recorded for geographical analysis in case is needed. The data is stored into a relational database so that it can be exported easily to other formats.

C. Automated Analysis of Usability Data

Despite longstanding research in data extraction and mining, there is a dearth of automated methods for usability evaluation based on user interaction traces. The described approach falls under the category of benchmark usability test. A set of benchmark tasks are predefined within the task based descriptor which is created by a usability expert to describe the goals to be achieved by the user. The task model is thereafter compared to the collected usage data which includes all user actions such as recorded mouse clicks and cursor movement. The matching process is based on well-defined and conventional metrics that reflect better usability. The chosen metrics are chosen on the basis they can be quantified automatically without the cooperation of the participants. The usability metrics considered in this study include:

- *Time spent per task* : is defined as the total time taken to achieve a particular task by a user. This metric is usually used to measure the efficiency rate. Based on the task descriptor, task duration is approximated through a sequential search within the user traces for the *Success* condition being met corresponding to the defined task.
- *Completion rate*: is also called the success rate which is considered one of the most fundamental usability metrics. The completion rate is typically measured as a binary value for task success (coded as 1) or task failure (coded as 0). Although, it is possible to define criteria for partial task success, but for simplicity reasons, binary values are considered. This is estimated in the same way as the task duration.
- *Mouse Clicks and Movement* : This is to measure the efforts undertaken by the user reflected through the use of hand to moving or clicking the mouse. In practice, larger number of clicks or longer distances of the cursor are indication of poor usability and lower satisfaction level.
- *Errors:* defined as Unintended actions or fail actions made by a user while doing a task in order to attempt a specific goal. The automated process for discovering error is to search the data log for non-compliance against the task-descriptor. This provides a good criteria to evaluate usability of the interactive system and infer the correlation between the user and task model.

In fact, recent studies [9] have proposed to produce a usability index. However, we believe that producing a number that reflect the degree of good usability is a complex and intricate task that should involve many factors. However, automated evaluation can be achieved through statistical analysis of data measuring the intra-correlation of estimated metrics for collected data against optimal data by an expert. Statistical analysis can be sufficient to identify major usability problems. This includes cases where there is higher variance of metrics among users.

IV. Experimental Results

In order to explore the effectiveness of the proposed approach for evaluating usability of web applications, experiments are conducted using a newly developed website where users are invited to use the site remotely. The web application is an interactive online quiz containing questions related to the tourism sector for the City of Souk Ahras. The task descriptor is made to contain 4 consecutive tasks. During the first task, the user is presented with a landing welcome page containing a button to start the quiz. Subsequently, participants would be taken throughout 5 different questions with a single question on each page. Choices of multiple answers are provided with each question. Thirdly, the user is taken to a page to show them the score they have attained when answering the questions with a button to continue the quiz. In the last task, a form asking anonymously the user for personal information such as gender, age range and their opinion regarding how easy to use the website. The script for logging user activities is hosted on an Amazon Cloud Services EC2 to account for faster access. For legal and privacy concerns, users are being told in advance that their traces are recorded for improving user experience and analyzing website usability.

During the usability evaluation process, 44 participants agreed to take part of the experiment. The rest of users did not want to disclose their gender and age. Upon testing the application, users are not required to install any software apart from using their preferred browser to test the interface. All actions and events performed by the users are recorded automatically and non-intrusively into the log data-set. To

assess the usability evaluation, the discussed metrics are computed automatically based on reading the task descriptor and user traces. Metrics include number of clicks, duration and cursor distance. This is computed individually for every higher level task.

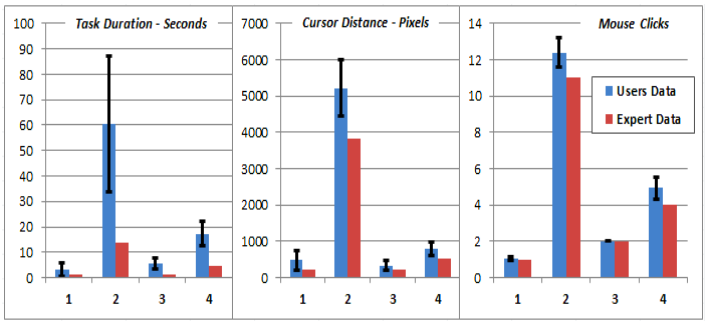

Fig.3. Estimated Usability Metrics for the four Tasks

Figure (3) shows the summative results obtained based on the derived metrics for the four tasks. The user data is estimated as the mean of measurements derived automatically of all participants for the three shown dimensions: Task duration, cursor distance and mouse clicks. The error bars in the plot on the users data correspond to the standard deviations of the measurements. It is observed that there is always a considerable gap between the expert and users logged data with the expert having always lower values compared to the average user. For the case of task 2, there is a high variance among users in terms of time in addition to the fact that there is a remarkable difference between the expert and users which is the same for task 4. This can be an indicative to a usability drawback of the designed interface at this phase of the application that needs to be addressed. Conversely, the number of clicks seems to be consistent between the two parties for most of the cases.

V. Conclusions

Usability which concerns the easiness of use for interactive systems, is recognized as an important software quality attribute, earning its place among more traditional attributes. Because of the scarce nature of methods devoted to the automated evaluation of the usability of web applications, this research study is carried out to demonstrate the use of a newly proposed task descriptor for automated remote evaluation. Empirical data recorded from end-users participating in a case study shows that usability metrics can be easily derived and analyzed to infer further insights about the usability of interactive systems.

References

[1] M. Christine Roy, O. Dewit, and B. A. Aubert, “The impact of interface usability on trust in web retailers,” Internet research, vol. 11, no. 5, pp. 388–398, 2001.

[2] M. Y. Ivory and M. A. Hearst, “The state of the art in automating usability evaluation of user interfaces,” ACM Computing Surveys, vol. 33, no. 4, pp. 470–516, 2001.

[3] A. Fernandez, E. Insfran, and S. Abrah˜ao, “Usability evaluation methods for the web: A systematic mapping study,” Information and Software Technology, vol. 53, no. 8, pp. 789–817, 2011.

[4] T. Tullis, S. Fleischman, M. McNulty, C. Cianchette, and M. Bergel, “An empirical comparison of lab and remote usability testing of web sites,” in Usability Professionals Association Conference, 2002.

[5] L. Paganelli and F. Patern`o, “Intelligent analysis of user interactions with web applications,” in International conference on Intelligent user interfaces, 2002, pp. 111–118.

[6] F. Patern`o, C. Santoro, and L. D. Spano, “Improving support for visual task modelling,” in Human-Centered Software Engineering. Springer, 2012, pp. 299–306.

[7] T. Tiedtke, C. M □artin, and N. Gerth, “Awusa–a tool for automated website usability analysis,” in Workshop on Interactive Systems. Design, Specification, and Verification. Rostock, Germany June, 2002, pp. 12–14.

[8] R. Atterer and A. Schmidt, “Tracking the interaction of users with ajax applications for usability testing,” in Pro- ceedings of the SIGCHI conference on Human factors in computing systems. ACM, 2007, pp. 1347–1350.

[9] L. G. de Vasconcelos and L. A. Baldochi Jr, “Towards an automatic evaluation of web applications,” in Proceedings of the 27th Annual ACM Symposium on Applied Computing. 2012,

[10] K. Muhi, G. Sz □oke, L. J. F □ul □op, R. Ferenc, and A. “A semi-automatic usability evaluation framework,” in Computational Science and Its Applications–ICCSA 2013.

[11] S. Andrica and G. Candea, “Warr: A tool for high-fidelity web application record and replay,” in 41st International Conference on Dependable Systems & Networks, 2011.

[12] I. Bouchrika, L. Ait-Oubelli, A. Rabir, and N. Harrathi. "Mockup-based navigational diagram for the development of interactive web applications." In Proceedings of the 2013 International Conference on Information Systems and Design of Communication, pp. 27-32. ACM, 2013..

[13] B. E. John and D. E. Kieras, “The goms family of user interface analysis techniques: Comparison and contrast,” ACM Transactions on Computer-Human Interaction (TOCHI), vol. 3, no. 4, pp. 320–351, 1996.

[14] N. A. Stanton, “Hierarchical task analysis: Developments, applications, and extensions,” Applied ergonomics, vol. 37, no. 1, pp. 55–79, 2006.

[15] P. Harms and J. Grabowski, “Usage-based automatic detection of usability smells,” in Human-Centered Software Engineering. Springer, 2014, pp. 217–234.

[16] S. Caffiau, D. Scapin, P. Girard, M. Baron, and F. Jambon, “Increasing the expressive power of task analysis: Systematic comparison and empirical assessment of tool-supported task models,” Interacting with Computers, vol. 22, no. 6, pp. 569–593, 2010.

[17] L. T. Long, N. T. Binh, and I. Parissis, “A new test modeling language for interactive applications based on task trees,” in Proceedings of the Fourth Symposium on Information and Communication Technology. ACM, 2013, pp. 285–293.

[18] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse, “The lumiere project: Bayesian user modeling for inferring the goals and needs of software users,” in Conference on Uncertainty in artificial intelligence, 1998