DOI: http://dx.doi.org/10.1016/j.jvlc.2014.12.002

Empirical validation of an automatic usability evaluation method

Rosanna Cassino, Maurizio Tucci, Giuliana Vitiello[*], Rita Francese

Department of Management and Information Technology, Università di Salerno, Italy

Keywords:
Interactive systems evaluation
Usability engineering
Summative usability evaluation

A B S T R A C T

Today, the success of a software application strongly depends on the usability of its interface, so the evaluation of interfaces has become a crucial aspect of software engineering. It is recognized that automatic tools for graphical user interface evaluation may greatly reduce the costs of traditional activities performed during expert evaluation or user testing in order to estimate the success probability of an application. However, automatic methods need to be empirically validated in order to prove their effectiveness with respect to the attributes they are supposed to evaluate.

In this work, we empirically validate a usability evaluation method conceived to assess consistency aspects of a GUI with no need to analyze the back-end. We demonstrate the validity of the approach by means of a comparative experimental study, where four web sites and a stand-alone interactive application are analyzed and the results compared to those of a human-based usability evaluation. The analysis of the results and the statistical correlation between the tool's rating and humans' average ratings show that the proposed methodology can indeed be a useful complement to standard techniques of usability evaluation.

1. Introduction

The most intuitive definition of usability is the property of the system that defines its degree of simplicity of use in terms of learning, storage and efficiency. The ISO 9241 standard, on “Ergonomics of Human System Interaction”, defines usability as “The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.”

From 1980 the value of a software system is measured on the basis of its GUIs (graphical user interfaces) and the related power of expression and communication. The interface has to be user-friendly because it is often the only part of the system with which the user interacts [20].

To create a “usable” system, the designer must create a good conceptual model of the application (correct, consistent, and simple) and effectively transmit it to the user through the interface that must “accommodate” user's mental model, namely her expectations about system behavior. Several evaluation methods and tools are available to measure to what extent a GUI is “usable”.

Usability inspection methods involve usability experts and different techniques (cognitive walkthrough with task-specific, heuristic evaluation, and pluralistic walkthrough) to evaluate a user interface without involving end users. These approaches, generally, can be used early during the development process by evaluating system prototypes or specifications that cannot be tested on end users [18].

Empirical methods refer to usability testing used in user-centered interaction design to evaluate a product by

E-mail addresses: rcassino@unisa.it (R. Cassino), mtucci@unisa.it (M. Tucci), gvitiello@unisa.it (G. Vitiello), francese@unisa.it (R. Francese).

testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system [4].

Such approaches detect usability deficiencies of the graphic interface by running and inspecting test cases and/or by analyzing the results of questionnaires or interviews. In addition to being very expensive and laborious, they often produce results that can be biased by the acquisition method and considerably depend on the adopted definition of usability, on the type and number of tasks, on the data and on the evaluation standards.

On the contrary, the automatic evaluation techniques are designed to avoid those problems both in terms of cost and in terms of running time: an automatic tool is able to locate in minutes (if not seconds) many critical issues. To get the same results with the above methods would take many hours of interviews and simulations of use cases. A comparative study carried out in the field confirms that automated tools for analytical evaluation are very efficient in terms of execution time, objectivity and reliability of the results obtained [11]. They can help designers to understand what usability problems may arise and how the interface should be improved, with respect to a given set of guidelines. However, most existing methods rely on interface source code to discover usability problems and offer advice on how to fix them.

In [1] we presented USherlock, a tool of GUI usability evaluation based on a front-side approach to derive the structure of the interface starting from what the user sees on the display. It allows to determine the nature of the elements of a graphical interface according to the changes produced on the interface itself. Depending on the type of interaction (input/output; pause; double click; click and double click; insertion of a character) and in case a visual feedback is found, the tool identifies some typologies of “dynamic” elements: button, link, text area, etc., and adds a new node to a tree which represents the hierarchical structure of each frame of the interface. For each element or set of elements classified the evaluation process will run all the usability controls. At the end of the evaluation process, each node is assigned a list of the inconsistencies identified and a score (rating between 0 and 1), which indicates the “quality” of the node.

In this paper, we present the results of the empirical study we have conducted to validate the proposed automatic usability evaluation technique with respect to its effectiveness, compared to the results of a traditional test, as it is usually performed during summative evaluation. We wanted to verify whether USherlock could be reliably adopted to evaluate usability of given artefacts with an acceptable error coverage. We decided to compare its performance with the outcomes of a heuristic evaluation performed by evaluators with a medium level of expertise. Therefore, the participants we recruited for our experiment were 24 students from the graduate course of Computer Science, who had successfully passed the exam of HCI and Software Usability and were well-trained on heuristic evaluation techniques. We analyzed four typologies of web sites and a stand-alone application. A control group of four expert evaluators (two external independent evaluators and two of the co-authors) was in charge of providing data on the real usability issues characterizing the analyzed artefacts. The effectiveness of each usability evaluation method could then be measured as the ratio between the number of detected problems and the number of real problems (as specified by the control group).

The results of the empirical study show that a statistically significant improvement in terms of effectiveness is indeed achieved using USherlock with respect to heuristic evaluation performed by evaluators having a medium level of expertise, with obvious advantages in terms of time and cost with respect to the canonical manual tests. To show that the improvements are unlikely to have occurred by chance, we have applied a one-tail t-test with a significance p-value <0.05 on the collected results for the pair of usability evaluation methods.

A major lesson learnt from this research is that the actual adoption by practitioners of an automatic usability evaluation technique can be greatly encouraged if a rigorous empirical analysis is performed to prove its reliability in terms of problems discovered. The automatic detection of specific usability issues, with the derived advantages in terms of time and cost, can then be effectively combined with other forms of usability evaluation, such as user testing and heuristic evaluation, meant to measure other aspects of usability especially related to users' degree of satisfaction.

The rest of the paper is organized as follows. Section 2 discusses about some related work. Section 3 summarizes the architecture of the implemented tool USherlock and describes some algorithmic aspects. In Section 4 we describe the empirical evaluation process developed to validate the tool analysis results. Section 5 contains some conclusions and final remarks.

2. Related work

The field of usability evaluation has been widely investigated over the last years. Several methods have been proposed both for the evaluation of web applications and of interactive graphical applications. However, many of such methods have failed to meet practitioners' expectations with respect to their usability evaluation goal, resulting in a low adoption rate. The need has therefore emerged among HCI researchers to perform appropriate studies meant to evaluate the usefulness of a usability evaluation method and the benefits gained by its adoption. The importance of rigorous studies to validate usability evaluation methods was first claimed by Gray and Salzman in [6]. The authors reviewed several experimental studies performed on usability evaluation methods and concluded that most suffer from the lack of meticulousness in proving the statistical validity of the achieved results. Similar claims were later discussed by Hartson et al., who also addressed the lack of standard criteria and a clear understanding of the factors being measured as major problems when comparing different usability evaluation methods [8]. A few years later Hornbæk summarized current practice of measuring usability and critically reviewed that practice, after reviewing usability measures employed in 180 studies published in core HCI journals and proceedings [9]. The goal of their study was to understand what challenges should be faced to improve the validity and reliability of usability measures. In the specific field of web engineering, Fernandez et al. presented an extensive analysis of the usability evaluation methods employed over the last years to evaluate web applications [5]. Their systematic study was mainly intended to examine how the analyzed usability evaluation methods relate to the web development process. Recently, the same researchers have presented the results of the empirical validation of a usability inspection method for model-driven web development, comparing it with heuristic evaluation performed in early stages of web development [4]. As a result of the qualitative analysis performed, they conclude that, given the nature of the proposed Web Usability Evaluation Process (WUEP), which relies on different metrics to be collected throughout the development process, most of the calculation tasks could be profitably automated.

Hvannberga et al. proposed a framework to compare different evaluation methods and described a case study where the framework is used to evaluate effectiveness, efficiency and inter-evaluator reliability of two sets of heuristics, Nielsen's heuristics and the cognitive principles of Gerhardt-Powals [10]. The same experiment was used to compare paper based methods and web tool based methods for usability problem reporting. The results suggest that a web tool for easy access and effective management of data captured during evaluation may yield more reliable results, especially supporting collaboration among evaluators in remote settings [9]. Similar claims are supported by Palanque et al., who described an approach that combines different techniques including formal analysis of models, simulation and analysis of log data in a model-based environment [19]. The log data at model level can be used not only to identify usability problems but also to identify where to operate changes to these models in order to fix usability problems. Some researchers have proposed the use of formal grammar specifications to support the designer in modeling and code generation of the GUI while satisfying usability parameters. In [2], a methodology was presented to specify and evaluate interactive visual environments, in particular web interfaces, based on the SR-Action Grammars formalism. A bottom-up approach is described to aid the designer to develop graphical applications that automatically respect a significant number of usability rules before the software is released and tested by standard methods, by means of checks performed at a high level of abstraction. However, similar solutions require a deep knowledge of the formalism to perform usability checks of the application, and are not conceived to verify applications already developed using a top-down approach.

A common aspect of all the previous studies is that they focus on usability evaluation methods mainly applied at early stages of design and development for formative usability evaluation activities. Such methods are therefore meant to improve the evaluated artefact before the implementation phase. The empirical study we present in the paper is instead referred to an automatic evaluation method intended for a summative usability evaluation.

Ivory and Hearst investigated the level of automation offered by a large number of existing WIMP and Web interfaces usability evaluation methods and discovered that some methods successfully automate aspects of usability evaluation such as capture, analysis, or critique activities [11]. They claim that automation of usability evaluation, as a proper complement to standard evaluation techniques, may yield several advantages. It may reduce time and cost required for evaluation activities, may increase consistency of uncovered errors, may help predicting time and error costs across design, may reduce the level of expertise required to evaluators, and may increase the coverage of evaluated features.

Several techniques have been explicitly conceived for automatic discovery of usability faults. Reverse engineering techniques have been implemented in order to extract static properties of the graphical interfaces, semi-automatic tools are then delegated to analyze the properties of the several GUI components [1,7,12-16]. To evaluate the efficiency of the interactive aspects, most of the systems extract test cases of the typical activities and tester users are assigned to the evaluation.

A number of methods for usability evaluation rely on the availability of the source code of an application and/or its interface. The process described in [3] analyzes the source code of a visual application, the design model of its interface, the graphical aspects and the interaction mechanisms implemented in each frame, panel and/or page of the system, to produce a report of the quantitative evaluation of heuristic factors.

One limitation of the automated tools previously examined is that they use a back-side approach, which derives from the analysis of the “hidden” side (e.g., source code and logs) of the system. If the source code is not available, it is very difficult to perform an automatic evaluation by means of the techniques above.

With reference to our present research goal, the revised literature reveals that the efficacy of the mentioned techniques has been primarily verified through testing activities, observing the effects of their adoption on real interface prototypes, whereas rigorous empirical validation procedures that could strengthen and generalize the testing outcomes are still missing. The results of the empirical validation we present are intended to be a first step towards bridging that gap.

3. USherlock: an automatic usability evaluation tool

In this section, we summarize the features of the automatic evaluation USherlock tool proposed in [1,2]. The basic idea is to use a front-side approach to derive the structure of the interface starting from what the user sees on the display. An automatic system recognizes and analyzes the static and dynamic aspects of the interface. The system is fully integrated and does not require any manual intervention either during the identification of graphical components or during the evaluation process.

The tool automatically evaluates several of the most popular usability metrics for the design of usable user interfaces [17] (Nielsen heuristics). We derived a list of 15 properties related to the usability of the visible aspects of an interactive application.

- Aspect ratio. The ratio between the height of a window and its width. The numbers in the range from 0.5 to 0.8 are acceptable.
- Widget nesting. A high level of nesting implies an incorrect design and excessive use of containers. The widgets such as buttons, labels and text-area should not exceed the third level of nesting.
- Relationship between background and widgets. The ratio between the non-widget and the total area of the interface, expressed in percentage. High numbers (> 80) interfaces are "thin", low numbers (< 30) show interfaces too "heavy".
- Widgets density. The number of widgets divided by the total area of the screen (multiplied by 100,000 to carry out normalization). Numbers greater than 100 indicate that a relatively large number of widgets are present in a too small area.
- Widgets deployment. A measure of how equally distributed the widget on the screen. Two values are considered: one to measure the horizontal balance (the ratio between the total area of the widget in the left half of the screen and the total area of the widget in the right half) and the vertical equilibrium, which compares the distributions of widgets in the upper and in the lower areas of the interface. Higher values, between 4 and 10, show unbalanced screens. A limit value of 10 indicates empty or nearly empty areas (for example, a dialog box which has a single button).
- Margins between widgets. A measure of the margin between a widget and another and between a widget and the edge of the screen. Values below a certain level (i.e., 5 pixels or 1/10 of the widget size) are considered inappropriate. In addition, children of the same container widget should have the same margins.
- Widgets alignment. All nodes, children of the same node ("sibling" nodes), must have at least one coordinate in common among those of the 4 possible corners.
- Widgets color. Color distribution (histogram), applied only to the widget, with the exception of the pictures (if possible). A good deployment must contain 2 or 3 peaks: a greater number of peaks or a flat histogram indicate the use of an excessive number of colors.
- Background color. Color distribution (histogram), applied only to the non-widgets area. A good deployment must bear only one peak of color: a greater number of peaks or a flat histogram indicate that the interface has an unclear background.
- Easily recognizable edges in widgets. Widget free edges are acceptable only if they have a very high contrast with the background (in the case of the Euclidean distance at least greater than 100).
- Widgets shape and size. The widgets that are contained in the same physical or logical container (sibling nodes in the tree hierarchy) should have similar values of size, shape and aspect ratio (same order of magnitude or ratio close to 1).
- Clear and recognizable buttons. Each button of the interface must be clearly recognizable in terms of contrast, particularly with respect to the color of its parent node (in terms of Euclidean distance, at least 200). In addition, each node button must demonstrate a clear change of state on mouse-over.
- Permitted actions and immediate feedback. Each input action should not lead to any inconsistent state or system failure. For every action there must be a clear and visible change. In the case of lengthy operations, the processing state of the system must be communicated to the user.
- Reversibility. At any time it should be possible to return to the previous state of the system.
- Enter text (input area). Text areas should have a single background color. The contrast between the entered text and the background must be very high (e.g., Euclidean distance at least greater than 300).

The evaluation process assigns each frame of the GUI a list of the identified inconsistencies and a score (ranging from 1 to 10) indicating its "quality".

To effectively perform the analysis of the previous properties of an interface, it is necessary to

- capture all the frames of the interface and cleaning;
- identify all the relevant elements;
- create a hierarchy of elements in an appropriate data structure;
- classify the elements according to their interactive features;
- evaluate each of the properties listed above;
- present a report with the obtained results.

Fig. 1 summarizes the flow of execution of the various tasks. The output of each task is the input for the next phase, even if in each phase it is possible to return to the previous task.

3.1. *Capture of the interface and cleaning*

The first task of the process is the acquisition of the interface, such as the URL address of the web page to be analyzed. A preliminary cleaning of the input is very important in order to exclude from further analysis all the irrelevant elements (e.g.: animations, advertisements, etc.). The resulting image will contain only the points that do not change their RGB components during a given slice of time, in order to exclude from subsequent analyzes animations in the interface that could confound the evaluation system and then produce unexpected and/or incorrect results (Fig. 2).

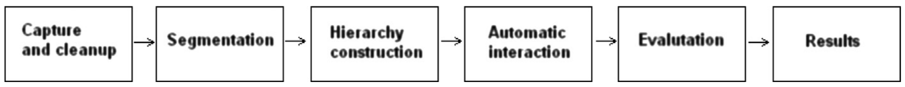


Fig. 1. Tasks execution flow.

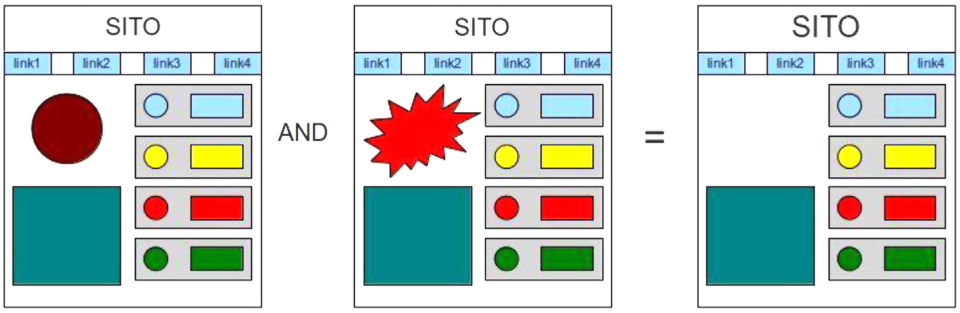


Fig. 2. Capture and cleaning task.

3.2. *Identification of elements by means of image segmentation*

The next step identifies the elements drawn on the interface such as images, text or widgets (buttons, menus, text areas, etc.) by an image segmentation algorithm that emulates the human visual perception [11].

The human visual perception alternates in the following steps:

- Environment input (visual stimulus);
- Register sensory (organs);
- Selective attention to interesting input;
- Perception and recognition.

The selective perception of humans refers to the Gestalt principles (where the German word Gestalt means form, pattern, representation) [19]. The main rules of organization of the data collected are

- Figure and Background. The relationship between figure and background can "read" the image, through the separation of figure from background. The dominant elements are perceived as figure, the rest is perceived as background.
- Proximity. It occurs when elements are close. In this case they are perceived as groups.
- Closure. It occurs when an object is incomplete or a space is not completely closed. Our eye tends to fill in the blanks and forms which are not closed. We tend to see the full image even when some information is missing.
- Continuation. It occurs when the eye is compelled to move through one object and continue to another.
- Similarity. Similar visual elements are grouped based on the shape, size, color or direction.
- Common destiny. It refers to objects that move in the same direction and therefore tend to be perceived as a single entity.
- Common region. Figure positioned within the same closed region tend to be perceived together.
- Symmetry. It consists in perceiving the whole of a figure through the individual constituent parts.

The segmentation step partitions an image into homogeneous regions on the basis of a certain criterion of membership of the pixels to a region.

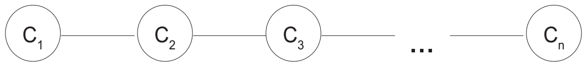


Fig. 3. Connections list.

The criteria for membership are

- Distinction: no pixel can be assigned to more than one region.
- Completeness: all the pixels of the image must be assigned at least one region of the partition.
- Connection: all the pixels belonging to a region must be "connected".
- Uniformity: all regions should be homogeneous with respect to a fixed criterion (e.g.: intensity, color, texture, etc.).

Among the various existing techniques, we chose to implement a SRM (Statistical Region Merging) algorithm [13]. SRM is a segmentation technique based on splitting and merging. The first splitting phase involves the subdivision of an image into a number of sub-regions, possibly composed initially of a single pixel. As a second step, a connection is created for each pair of adjacent regions to abstractly encapsulate the following information: a reference to the two connected regions, and the degree of similarity of the two regions. Then the algorithm creates a list of connections, ordered in an ascending order according to the similarity degree (Fig. 3).

The subsequent merging step simply performs the fusion of pairs of connected (adjacent) regions, if they are considered homogeneous on the basis of a comparison operator. The algorithm performs an iterative loop on the list of connections and at each step the system evaluates the connection in question: if the degree of similarity between the regions involved is greater than a predetermined threshold, the system blends together the regions concerned (phase merge). For example, the merger of a region A with region B will generate a new region D. We have used the Euclidean distance of the RGB components of the average color. For example, a region C that first appeared connected with A through a connection c_2 will now be connected with the new D region: this can in fact change the value of the degree of compatibility of the connection that involves the regions and to prevent, if the compatibility is below the fixed threshold, their relative fusion, unless additional updates (Fig. 4).

After the merging step, the SRM algorithm performs a further refinement of the results, which allows to clean up the areas of the image that are too small or of little interest: each small region is merged with the most similar neighboring region; to determine if a region is small it is possible to use a default value, or a parametric value (e.g., to relate the value to the size of the interface). The output of the segmentation process is a map of homogeneous regions, and the result is presented on the screen.

3.3. *Hierarchy of elements in an appropriate data structure*

The result of the segmentation process is a set of homogeneous regions. These regions correspond to the elements of the interface. The next step describes the relations between the elements of the GUI and their hierarchical composition as a tree data structure, in which one or more elements (children nodes) are contained in an element of type container (parent node) (Fig. 5). At first, all regions are inserted in a list and sorted according to their coverage area (i.e. the area of the minimum rectangle that contains all the pixels of the region). An iterative loop is performed on each region of the list. For each iteration of the main loop, a nested loop is performed to find a parent candidate region. If the surface of the examined region is fully contained within the surface of the compared region, the iteration stops and the examined region is included in the set of children nodes of the parent candidate region; otherwise the cycle continues repeating on the next parent candidate region. At the end, if there is more than one root, a dummy node is added as the new root of the tree and all nodes in the previous step become its children, so that a single tree structure is the result of the hierarchization process.

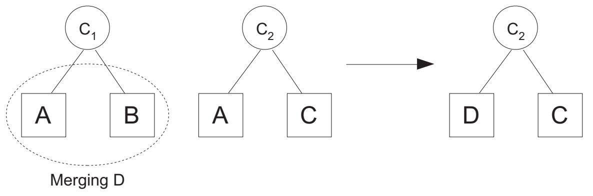


Fig. 4. Merge of regions.

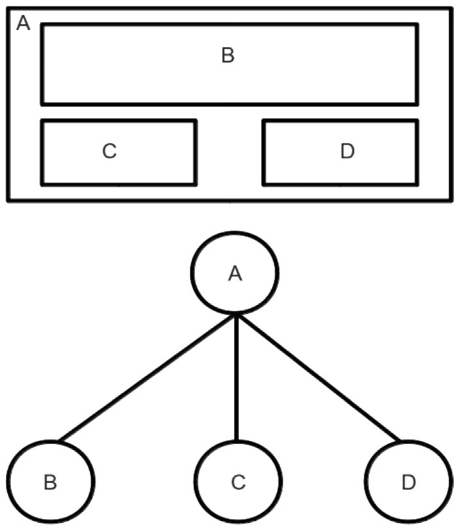


Fig. 5. Element hierarchy.

3.4. *Classification of elements and analysis of the interactive mechanisms*

The type (static or dynamic) of each identified object determines which aspects of the evaluation apply. To this aim, we have designed an automatic tool that determines the type of each element of an interface (label, button, link, text-area, menus, etc.) by means of the changes to the interface itself that are activated by interacting with it. To simulate interaction, the tool generates automatic input events, such as mouse moving and clicking, or key pressing and releasing.

For each element contained in the data structure, the system attempts to simulate an interaction with it, progressively generating the following input events:

– mouse input in the region;
– pause of the mouse over the region;
– exit of the mouse from the region;
– press and release (click) of the left button of the mouse;
– double click of the left button of the mouse;
– attempt to enter at least two characters of the alphabet.

For each generated input the system records any changes of the interface and determines the nature of the object. Depending on the type of input and on the

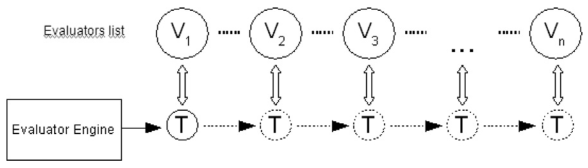


Fig. 6. Evaluator engine.

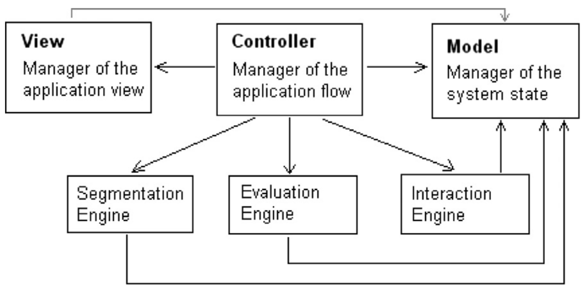

View
Manager of the application view
Controller
Manager of the application flow
Model
Manager of the system state
Segmentation Engine
Evaluation Engine
Interaction Engine

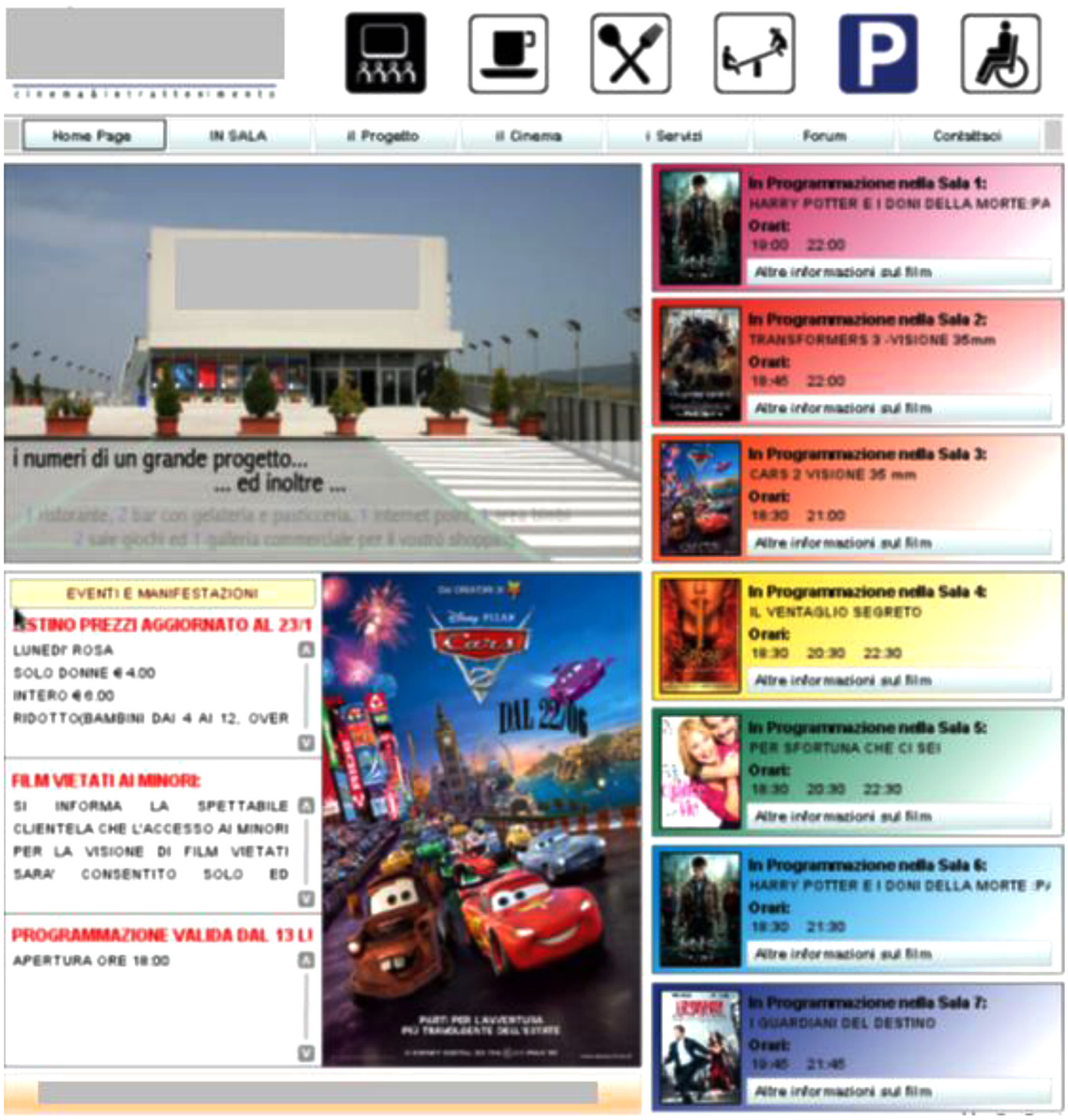
Fig. 7. The implementation model.

detected visual feedback, the following typologies of elements are determined:

– input/output: dynamic element (it is possible to interact with it);
– pause: the system provides a description of the item;
– press and release of left button of the mouse: the element is a button or a menu item;
– double click: if the visual feedback is isolated to a neighborhood of the point with which you interacted, the element is a character and all pixels affected by the change form a group (string);
– click and double click: the item is a link;
– insertion of a character: the element is a text field or a text-area.

As the classification of elements is completed, a further refinement operation improves, where possible, the structure of the tree hierarchy, by applying the following rules:

– identification of false hierarchies: regions linked by a parent–child relationship and showing the same behavior are blended together;



i numeri di un grande progetto...
... ed inoltre ...
1 ristorante, 2 bar con gelateria e pasticceria, 1 internet point, ...
2 sale giochi ed 1 galleria commerciale per il vostro shopping

LISTINO PREZZI AGGIORNATO AL 23/1
LUNEDI' ROSA
SOLO DONNE € 4.00
INTERO € 6.00
RIDOTTO(BAMBINI DAI 4 AI 12, OVER

FILM VIETATI AI MINORI:
SI INFORMA LA SPETTABILE CLIENTELA CHE L'ACCESSO AI MINORI PER LA VISIONE DI FILM VIETATI SARA' CONSENTITO SOLO ED

PROGRAMMAZIONE VALIDA DAL 13 LI
APERTURA ORE 18:00

In Programmazione nella Sala 1:
HARRY POTTER E I DONI DELLA MORTE PA
Orari:
19:00 22:00
Altre informazioni sul film

In Programmazione nella Sala 2:
TRANSFORMERS 3 -VISIONE 35mm
Orari:
18:45 22:00
Altre informazioni sul film

In Programmazione nella Sala 3:
CARS 2 VISIONE 35 mm
Orari:
18:30 21:00
Altre informazioni sul film

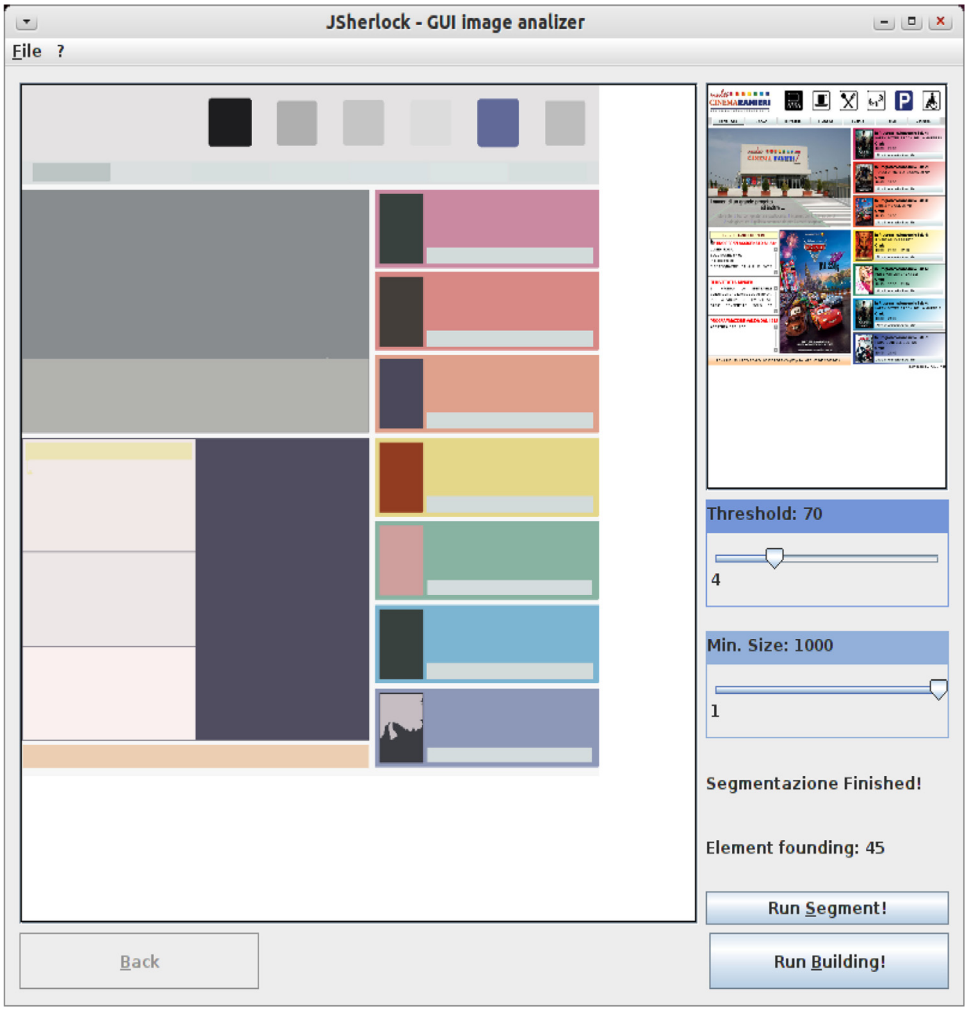
In Programmazione nella Sala 4:
IL VENTAGLIO SEGRETO
Orari:
18:30 20:30 22:30
Altre informazioni sul film

In Programmazione nella Sala 5:
PER SFORTUNA CHE CI SEI
Orari:
18:30 20:30 22:30
Altre informazioni sul film

In Programmazione nella Sala 6:
HARRY POTTER E I DONI DELLA MORTE PA
Orari:
18:30 21:30
Altre informazioni sul film

In Programmazione nella Sala 7:
I GUARDIANI DEL DESTINO
Orari:
19:45 21:45
Altre informazioni sul film

Fig. 8. Interface capture.



Segmentazione Finished!

Element founding: 45

Fig. 9. Interface segmentation.

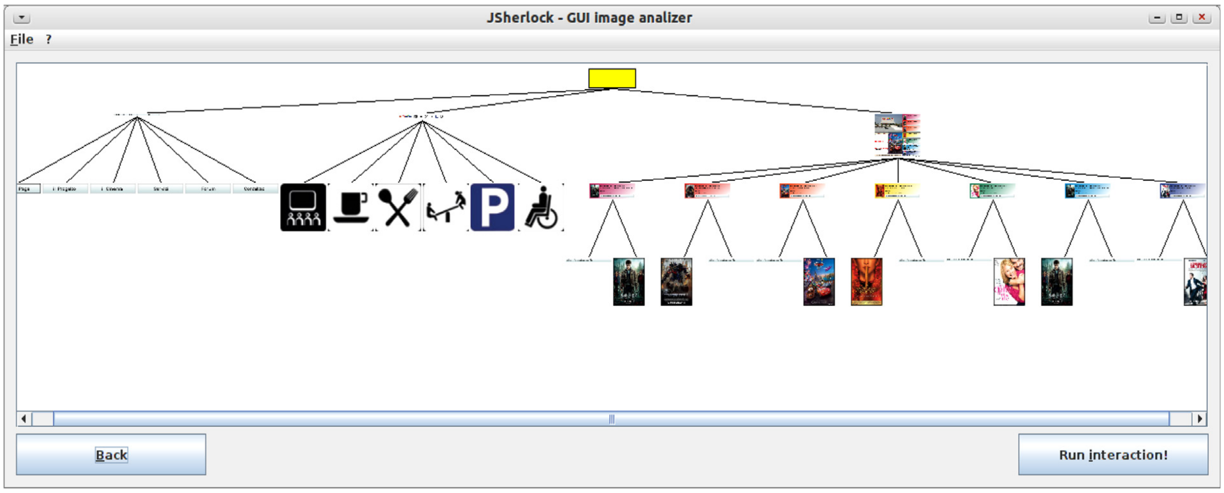
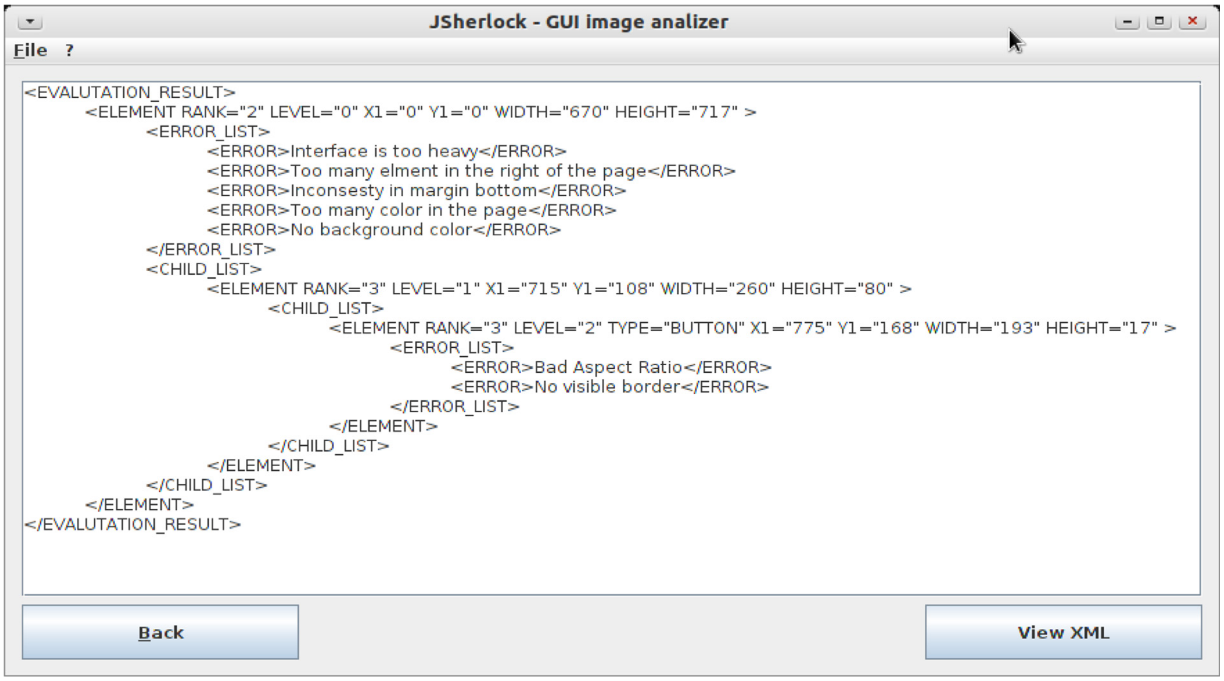


Fig. 10. The hierarchy tree.



```
<EVALUTATION_RESULT>
        <ELEMENT RANK="2" LEVEL="0" X1="0" Y1="0" WIDTH="670" HEIGHT="717" >
                <ERROR_LIST>
                        <ERROR>Interface is too heavy</ERROR>
                        <ERROR>Too many elment in the right of the page</ERROR>
                        <ERROR>Inconsesty in margin bottom</ERROR>
                        <ERROR>Too many color in the page</ERROR>
                        <ERROR>No background color</ERROR>
                </ERROR_LIST>
                <CHILD_LIST>
                        <ELEMENT RANK="3" LEVEL="1" X1="715" Y1="108" WIDTH="260" HEIGHT="80" >
                                <CHILD_LIST>
                                        <ELEMENT RANK="3" LEVEL="2" TYPE="BUTTON" X1="775" Y1="168" WIDTH="193" HEIGHT="17" >
                                                <ERROR_LIST>
                                                        <ERROR>Bad Aspect Ratio</ERROR>
                                                        <ERROR>No visible border</ERROR>
                                                </ERROR_LIST>
                                        </ELEMENT>
                                </CHILD_LIST>
                        </ELEMENT>
                </CHILD_LIST>
        </ELEMENT>
</EVALUTATION_RESULT>
```

Fig. 11. Evaluation results.

Table 1
User questionnaire.

Usability properties	Questions	Value
Widgets density	How would you rate the amount of elements within the page? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)?	
Widgets deployment	How would you rate the distribution of elements on the page (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)?	
Widgets color	What do you think of the color of the elements on the page (1: inadequate; 2: too strong relative to the background; 3: almost unrecognizable from the background; 4: good, 5: excellent)?	
Background color	How would you rate the background color of the page? (1: inadequate; 2: confused, 3: hardly recognizable compared to the other elements of the interface, 4: good, 5: excellent)	
Aspect ratio	How would you rate the proportionality of the window size with respect to the visibility of its contents? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	
Widget nesting	How would you rate the nesting of the elements, text boxes and labels in the page? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	
Relationship between background and widgets	How would you rate the degree of uniformity in the relationship between the number of elements on the page and the space reserved for them? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	
Margins between widgets	How would you rate the degree of distinction between the elements in a page, referred to their margins? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	
Widgets alignment	How would you rate the alignment of elements inserted within another element of the page? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	
Easily recognizable edges in widgets	How would you rate the degree of discernibility of the widgets with respect to the background color of the page? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	
Clear and recognizable buttons	How would you rate the degree of perception of buttons based on their color intensity? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	
Clear and recognizable buttons to mouseover	How would you rate the recognition of the buttons when they are crossed by the mouse arrow? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	
Permitted actions and immediate feedback	How would you rate the amount of non-active links found? (1: high, 2: middle-high, 3. Middle, 4: low, 5: very low)	
Reversibility	In case of wrong action, how would you rate reversibility to the previous state? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	
Enter Text	When inserting text inside the appropriate sections, how would you rate the contrast between text and background? (1: inadequate, 2: poor, 3: sufficient, 4: good, 5: excellent)	

– introduction of new hierarchies: all the sibling nodes that show a similar behavior are grouped together, hence introducing a new sub-hierarchy.

3.5. *Elements evaluation*

The *Evaluation Engine* is the tool component, which is in charge of checking for inconsistencies of the interface. The engine analyzes each node of the tree and controls a list of particular aspect of usability. If it finds a violation in a node, a message of inconsistency is added to it (Fig. 6). We have created 15 different evaluators: each of them implements one of the assessments designed in the analysis and listed in the previous section.

3.6. *Result presentation*

The results of evaluation are shown to the user in a simplified or detailed manner. The first form of presentation is an aggregated report that shows a numerical score between 0 and 1 for each element. For the root node, the score is computed using a recursive relationship that links the inconsistencies of the node to the score of its children-nodes. In particular

$$\nu(n)=\left(\frac{1}{4}\left(\frac{n.con-n.inc}{n.con}\right)+\frac{3}{4}\left(\frac{\sum\nu(c_i)}{n.child}\right)\right)$$

where $\nu(n)$ is the score of a node, $n.con$ represents the total number of evaluations performed on the node, $n.inc$ represents the number of inconsistencies of the node, $\nu(c_i)$ is the score of child node i, and $n.child$ represents the total number of children.

The score $v(r)$ of the root node is computed by the following formula:

$$\nu(r)=\frac{\sum\nu(c_i)}{n.child}$$

that represents the normalized total score assigned to the interface.

The second form of presentation is a list of the inconsistencies identified for each node of the hierarchy. For the ith element, the list presents the following information: the upper-left pixel coordinates, size (width and height), surface, medium color, dynamic nature, the list of listened events, inconsistencies, and evaluation degree.

3.7. *How USherlock works*

The system is named USherlock, after the famous program written in ANSI C by Mahajan and Shneiderman [13], as a tribute to one of the first historical attempts of automation in the field of usability evaluation of graphical user interfaces.

USherlock is implemented in Java, following the MVC (Model-View-Control) architectural pattern (Fig. 7). The view package includes the classes related to the rendering of the model. It sends the controller each user requests and allows the controller to select a particular view.

The controller package defines the flow of execution. Its classes map user requests into actions performed on the

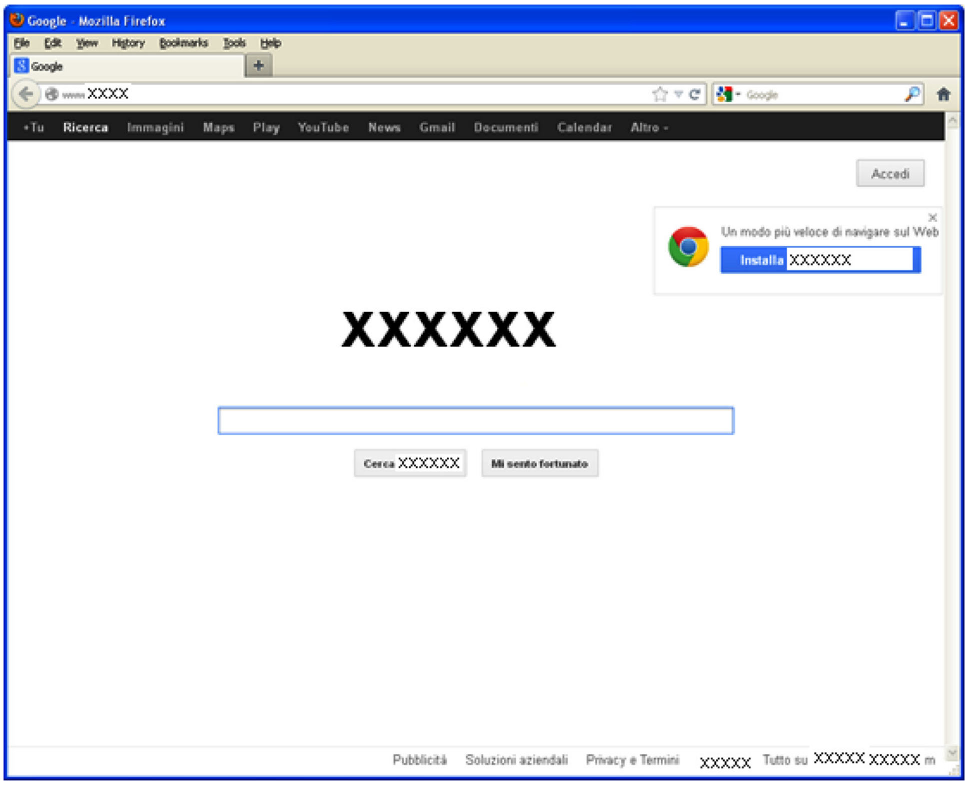

Google - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google
www.XXXX
Google
+Tu Ricerca Immagini Maps Play YouTube News Gmail Documenti Calendar Altro
Accedi
Un modo più veloce di navigare sul Web
Installa XXXXXX
XXXXXX
Cerca XXXXXX
Mi sento fortunato
Pubblicità Soluzioni aziendali Privacy e Termini XXXXX Tutto su XXXXX XXXXX.m

Fig. 12. The home page of a popular search engine.

model and select a view related to user requests. For each input event, a specific class describes the action to be performed on the model and selects the next view. The model package encapsulates the application state.

Fig. 8 shows an example of analysis of a home page of a web site. The input data required are the web page URL to be evaluated and the path of the browser. The *Interaction Engine* performs a series of 10 screenshots at intervals of one second, and then returns a list of captured images to the main process. Fig. 9 shows the result of the segmentation performed by the *Segmentation Engine*.

The *TreeBuilder Engine* designs the hierarchy of elements (Fig. 10). The analysis of the interaction mechanism of the examined interface is implemented as an automatic interaction process: an automatic interaction module simulates the input events generation, and a decision

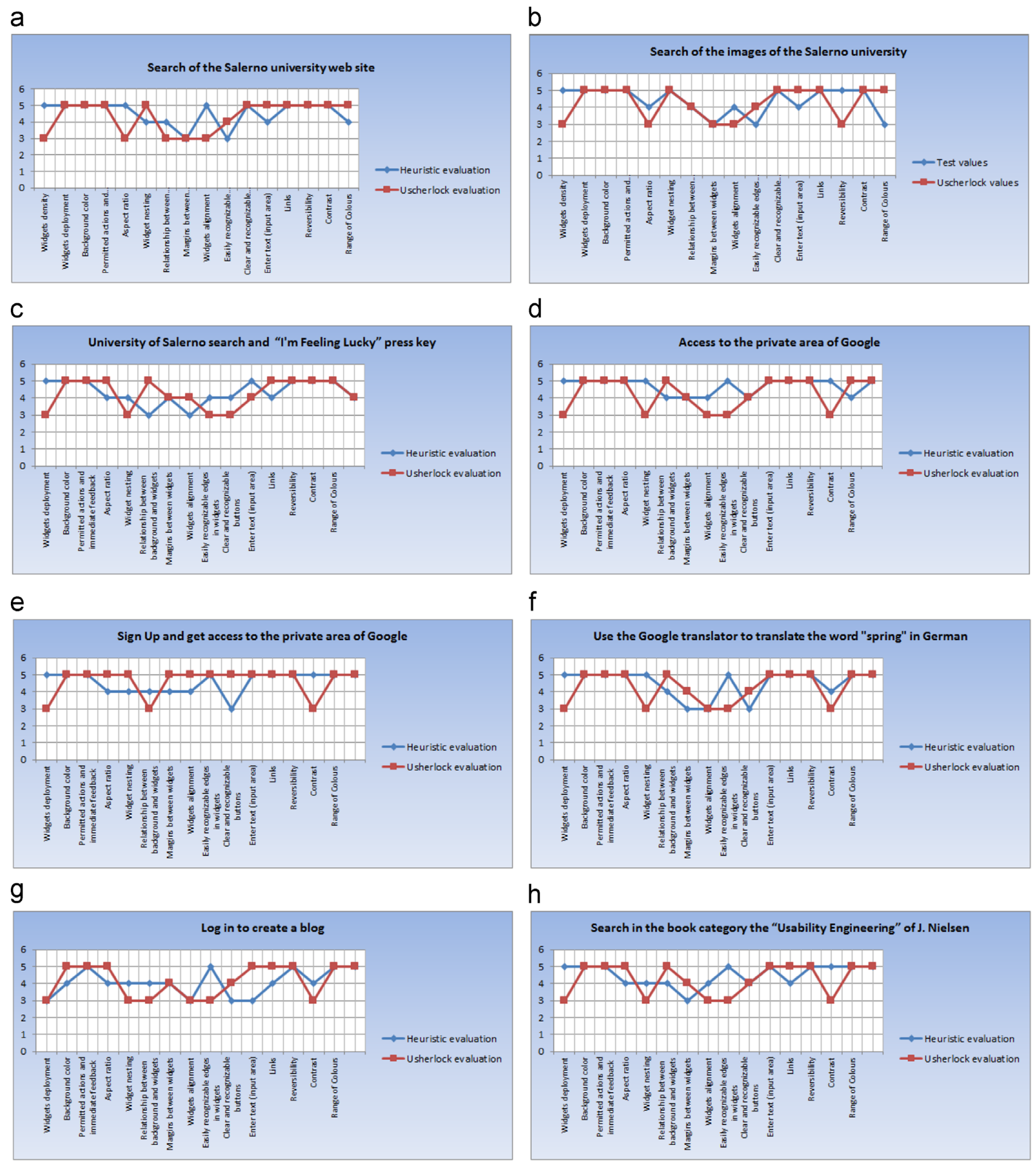


Fig. 13. The graphical results of the search engine analysis. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

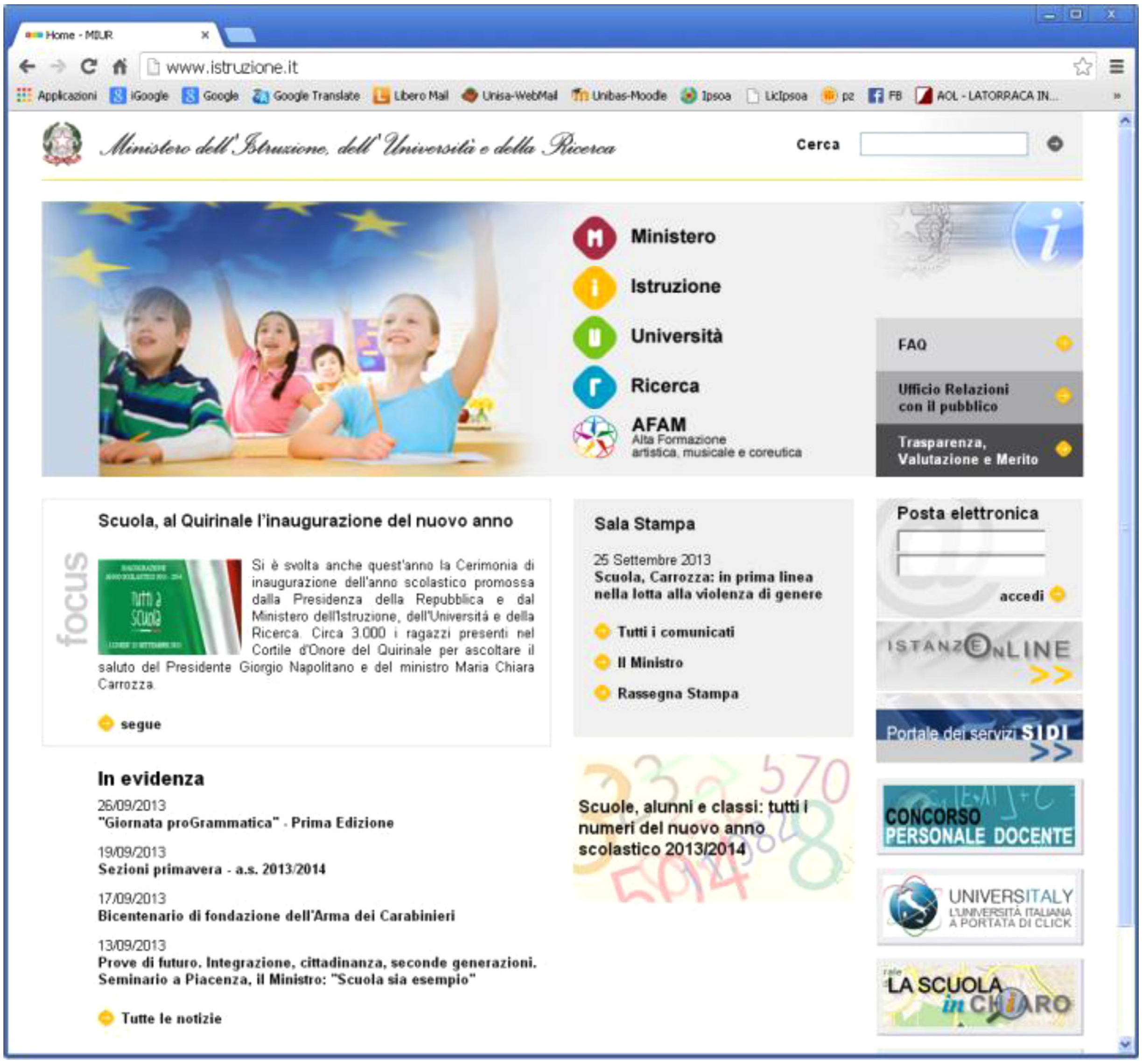
algorithm chooses the interface element to interact with (and what type of input to generate). The automatic interaction module is implemented using the Robot class of the java.awt package, whose methods allow the generation of system native input events. The decision algorithm is implemented using the *Interaction Engine*. The program performs a post-order visit of the tree and, for each object, asks the Robot to interact with it through the following events:

– input/output of the mouse in/out the region;
– click of the left button of the mouse on the median pixel of the region;
– double click of the left button of the mouse on the median pixel of the region;
– enter a character.

The *Evaluation Engine* performs the GUI evaluation of the inconsistencies using the 16 evaluators described in Section 3. Fig. 11 shows an example of the evaluation results.

4. Experimental analysis

The general concern addressed by the experimental analysis has been the evaluation of the validity of the implemented approach with respect to the usability issues detected by a traditional test, as it is usually performed during summative evaluation. In this section, we describe the study in terms of subjects, apparatus, variables and tasks, we explain the analytical methods employed to give evidence to our claims and discuss the achieved results.


Home - MIUR

www.istruzione.it

Applicazioni iGoogle Google Google Translate Libero Mail Unisa-WebMail Unibas-Moodle Ipsoa LicIpsoa pz FB AOL - LATORRACA IN...

Ministero dell'Istruzione, dell'Università e della Ricerca

Cerca

Ministero
Istruzione
Università
Ricerca
AFAM
Alta Formazione artistica, musicale e coreutica

FAQ
Ufficio Relazioni con il pubblico
Trasparenza, Valutazione e Merito

Scuola, al Quirinale l'inaugurazione del nuovo anno

focus

Si è svolta anche quest'anno la Cerimonia di inaugurazione dell'anno scolastico promossa dalla Presidenza della Repubblica e dal Ministero dell'Istruzione, dell'Università e della Ricerca. Circa 3.000 i ragazzi presenti nel Cortile d'Onore del Quirinale per ascoltare il saluto del Presidente Giorgio Napolitano e del ministro Maria Chiara Carrozza.

segue

Sala Stampa

25 Settembre 2013
Scuola, Carrozza: in prima linea nella lotta alla violenza di genere

Tutti i comunicati
Il Ministro
Rassegna Stampa

Posta elettronica

accedi

ISTANZE ONLINE

Portale dei servizi SIDI

In evidenza

26/09/2013
"Giornata proGrammatica" - Prima Edizione

19/09/2013
Sezioni primavera - a.s. 2013/2014

17/09/2013
Bicentenario di fondazione dell'Arma dei Carabinieri

13/09/2013
Prove di futuro. Integrazione, cittadinanza, seconde generazioni. Seminario a Piacenza, il Ministro: "Scuola sia esempio"

Tutte le notizie

Scuole, alunni e classi: tutti i numeri del nuovo anno scolastico 2013/2014

CONCORSO PERSONALE DOCENTE

UNIVERSITALY
L'UNIVERSITÀ ITALIANA A PORTATA DI CLICK

LA SCUOLA in CHIARO

Fig. 14. The home page of an institutional site.

4.1. *Goal of the experiment*

The goal of our experiment was to evaluate the effectiveness of the proposed automatic technique against a usability inspection method, performed taking into account the same properties related to the visible aspects of an interactive application. We also wanted to give evidence of its appropriateness for usability evaluators who have a medium level of expertise.

4.2. *The artefacts evaluated*

We analyzed four typologies of web sites and a stand-alone application. The chosen web sites cover a wide range of typical web interfaces, including a popular search engine, an institutional site, a site dedicated to the issues of usability and accessibility of websites, and a business site of information technology services. The stand-alone application is a commercial visual database for professionals (lawyers, labor

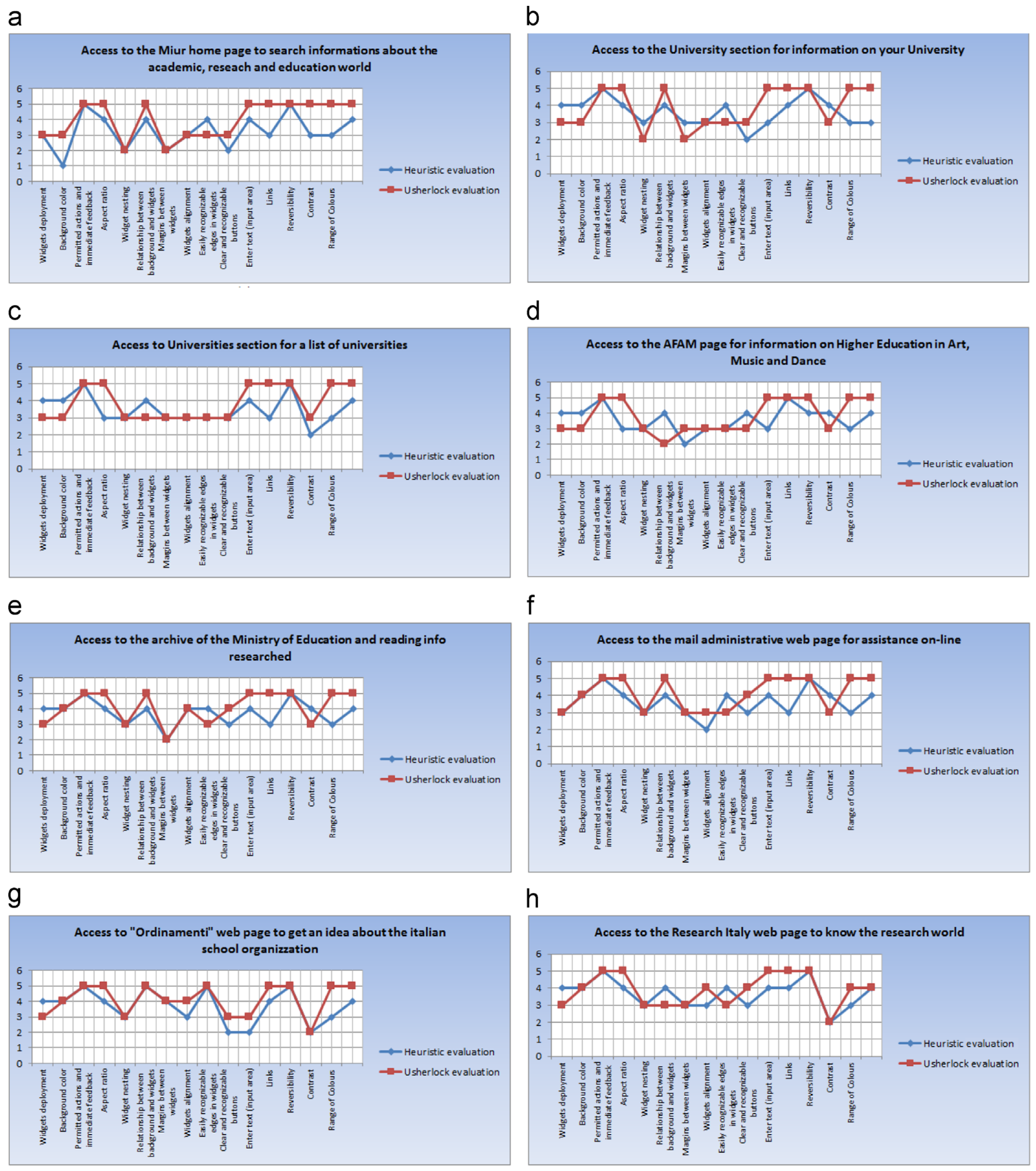


Fig. 15. The graphical results of the institutional site.

consultants, and accountants) that allows to search various types of documents and quick access to different types of papers (e.g. legislation, practice, and law). Although some of the interfaces are in Italian, the native language of evaluators, they are substantially identical to the more general English versions with respect to the usability indicators we considered for the study.

4.3. *Independent and dependent variables*

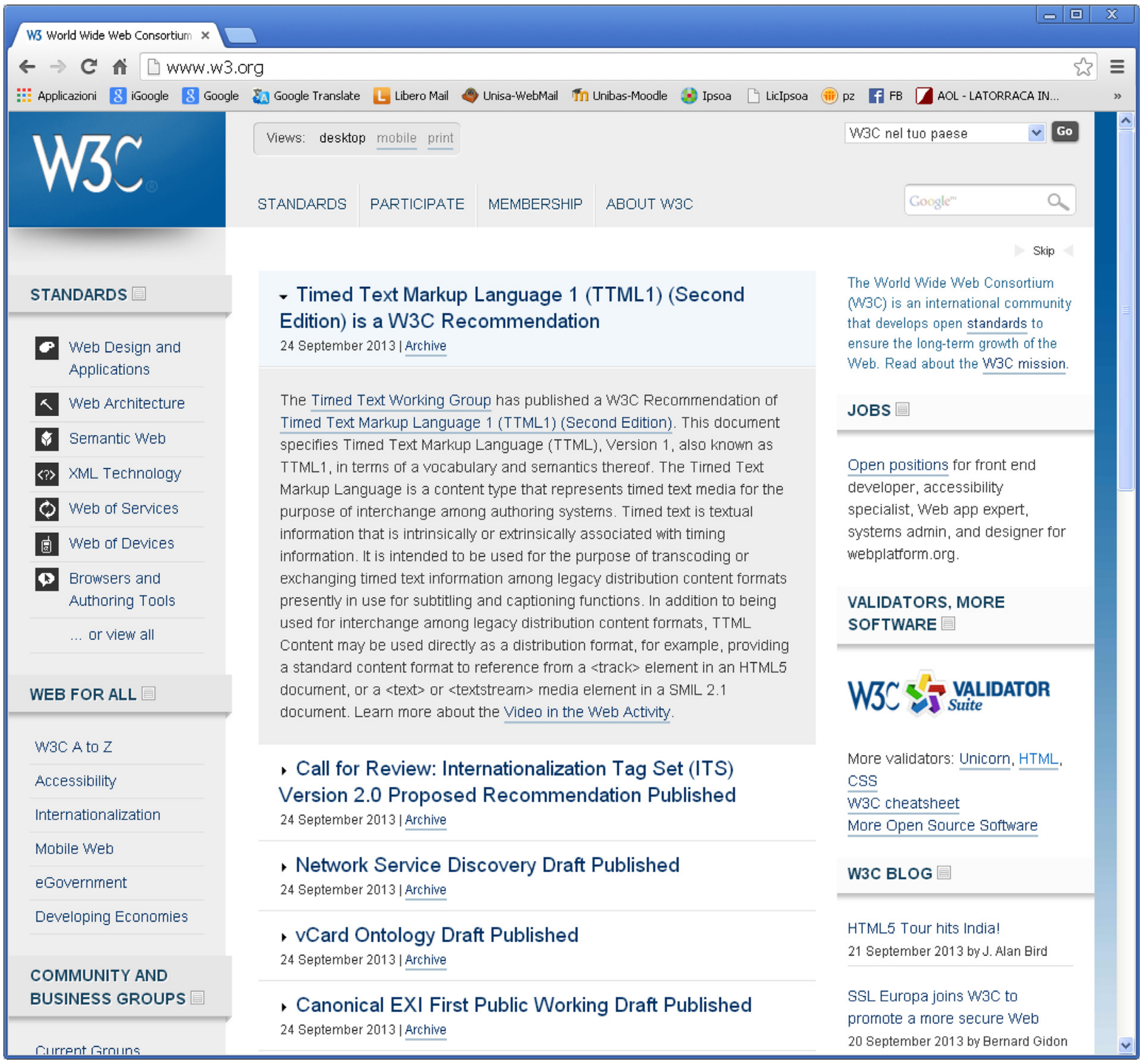
We identified the independent variable in our experiments is

- the evaluation method, with nominal values: USh (USherlock) and HE (heuristic evaluation).

The dependent variable is

- Effectiveness, which is calculated as the ratio between the number of usability problems detected and the total number of existing (known) usability problems, as discovered by a control group made up of expert evaluators. We consider one usability problem as one defect that can be found independently of its severity level and its total number of occurrences.

Following the approach given in [4], we set up a control group of four people, two independent expert evaluators and two of the authors, who worked independently from each other to identify usability problems with respect to the 15 usability properties analyzed adopting USh and HE methods, respectively. After separate evaluation sessions the control group members were asked to compare their analyses and agree on the set of real problems discovered, for each product analyzed. Possible biases deriving from



Timed Text Markup Language 1 (TTML1) (Second Edition) is a W3C Recommendation
24 September 2013 | Archive

The Timed Text Working Group has published a W3C Recommendation of Timed Text Markup Language 1 (TTML1) (Second Edition). This document specifies Timed Text Markup Language (TTML), Version 1, also known as TTML1, in terms of a vocabulary and semantics thereof. The Timed Text Markup Language is a content type that represents timed text media for the purpose of interchange among authoring systems. Timed text is textual information that is intrinsically or extrinsically associated with timing information. It is intended to be used for the purpose of transcoding or exchanging timed text information among legacy distribution content formats presently in use for subtitling and captioning functions. In addition to being used for interchange among legacy distribution content formats, TTML Content may be used directly as a distribution format, for example, providing a standard content format to reference from a <track> element in an HTML5 document, or a <text> or <textstream> media element in a SMIL 2.1 document. Learn more about the Video in the Web Activity.

Call for Review: Internationalization Tag Set (ITS) Version 2.0 Proposed Recommendation Published
24 September 2013 | Archive

Network Service Discovery Draft Published
24 September 2013 | Archive

vCard Ontology Draft Published
24 September 2013 | Archive

Canonical EXI First Public Working Draft Published
24 September 2013 | Archive

The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the Web. Read about the W3C mission.

JOBS

Open positions for front end developer, accessibility specialist, Web app expert, systems admin, and designer for webplatform.org.

VALIDATORS, MORE SOFTWARE

W3C VALIDATOR Suite

More validators: Unicorn, HTML, CSS
W3C cheatsheet
More Open Source Software

W3C BLOG

HTML5 Tour hits India!
21 September 2013 by J. Alan Bird

SSL Europa joins W3C to promote a more secure Web
20 September 2013 by Bernard Gidon

Fig. 16. The home page of the W3C site.

the subjective nature of the evaluation were therefore mitigated by their final brainstorming activity.

4.4. Participants recruitment

One of the goals of the present research was to verify whether evaluators with a medium level of expertise could reliably adopt USherlock to evaluate usability of given artefacts with an acceptable error coverage. Therefore, the participants we recruited for our experiment were 24 students from the graduate course of Computer Science, who had successfully passed the exam of HCI and Software Usability, were well-trained on heuristic evaluation techniques.

4.5. The experimental design

For each web site and for the stand-alone application we used USherlock to navigate and evaluate eight pages/screens. The same pages/screens were preliminarily submitted to the control group to get experts' ratings.

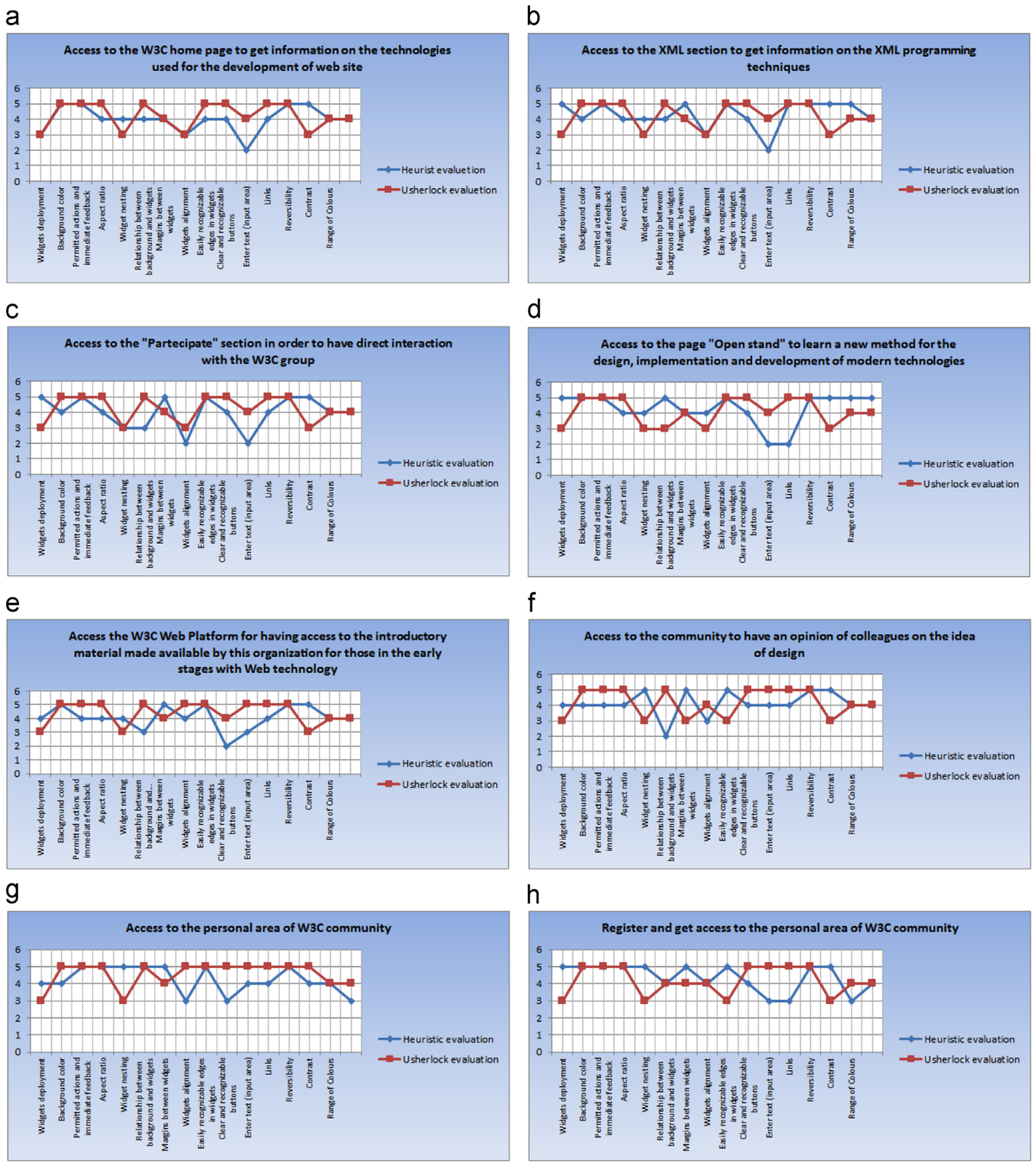


Fig. 17. The graphical results of any web page of the W3C.

The 24 students taking part in the experiment were initially informed on the purpose of the study and were instructed on the set of usability indicators considered relevant for the evaluation. They were then given instructions to perform one task for each of the analyzed pages/screens, with a total of 40 units of analysis. In order to avoid biases in the evaluation, participants were split into 4 separate groups of 6 people, randomly composed. To limit annoyance and loss of attention, four experimental sessions were run, on different dates. During a session each group focused on a different web site, cycling through the sessions, so that all groups would interact with the five evaluated artefacts.

At each session, group members performed the assigned tasks independently from one another, and upon completion of each task they were asked to fill in a questionnaire, composed of a set of 15 questions associated to a 1–5 rating scale. Table 1 shows the user questionnaire common to the analysis of all interfaces.

4.6. *The experimental tasks*

The first site examined is related to a popular search engine. Fig. 12 shows its home page. Participants were assigned the following tasks:

- *Search for the university of Salerno web site* (from https://www.google.it/webhp?hl=it&tab=ww)
- *Search for images of the university of Salerno (from* http://www.google.it/imghp?hl=it&tab=wi)
- *Search for the university of Salerno web site and press the "I am feeling lucky" button to reach directly the university homepage (from* http://www.google.it/ig?brand=SVED&bmod=SVED&aig=0&reason=1)
- *Sign in and access the reserved area* (from https://accounts.google.com/ServiceLogin?hl=it&continue=http://www.google.it/)
- *Register and access the reserved area* (from https://accounts.google.com/SignUp?service=mail&continue=https%3A%2F%2Fmail.google.com%2Fmail%2F%3Ftab%3Dnm&ltmpl=default)
- *Use the translator to find the German translation of the word "spring"* (from http://translate.google.it/?hl=it&tab=wT)
- *Sign in to create a blog* (from https://accounts.google.com/ServiceLogin?service=blogger)
- *Search the library for the book "Usability Engineering" by J. Nielsen* (from http://books.google.it/

The graphics depicted in Fig. 13 report the average scores assigned by participants to each task (the blue line in the graphics), compared to the evaluation results gained by the USherlock tool (red line).

Fig. 14 shows the home page of the second web site. The analyzed web pages are

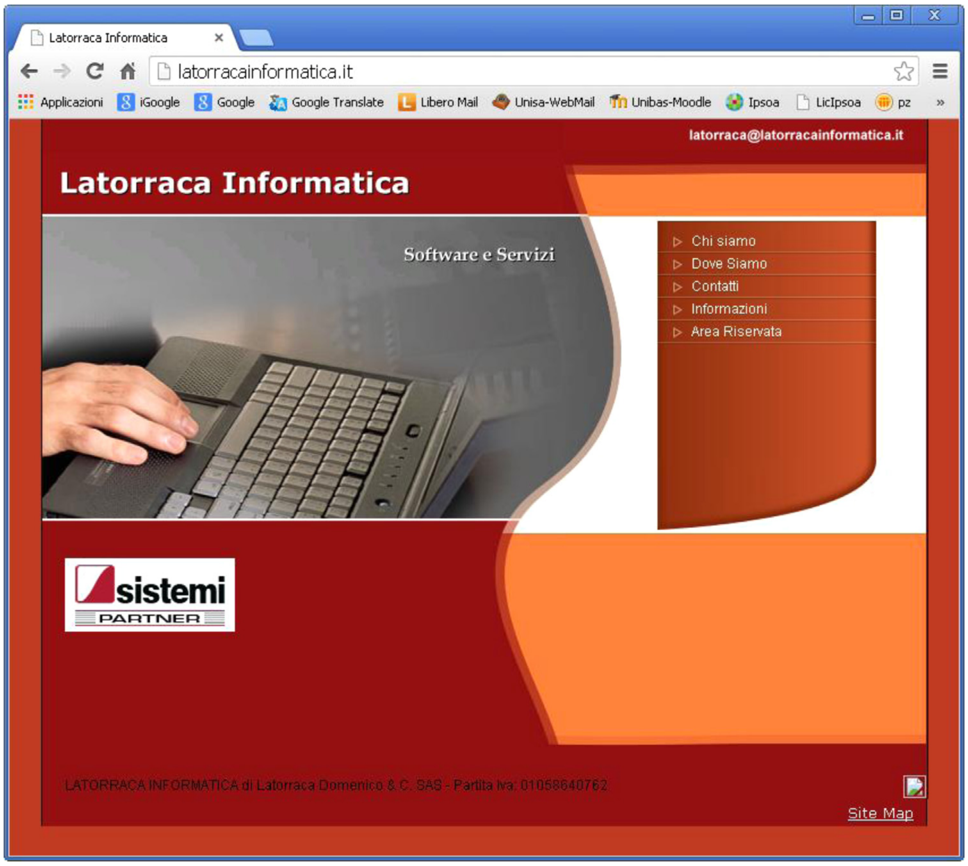


Fig. 18. The home page of the site of a business company.

- *http://www.istruzione.it/*
- *http://hubmiur.pubblica.istruzione.it/web/universita/home*
- *http://cercauniversita.cineca.it/index.php?module=strutture&page=StructureSearchParams&advanced_serch=1*
- *http://www.afam.miur.it/*
- *http://archivio.pubblica.istruzione.it/istanzeonline/index.shtml*
- *https://www.researchitaly.it/conoscere/*
- *http://archivio.pubblica.istruzione.it/webmail/posta_amministrativi.shtml*
- *http://hubmiur.pubblica.istruzione.it/web/istruzione/famiglie/ordinamenti*

Tasks and comparative graphics are depicted in Fig. 15. Fig. 16 shows the home page of the third web site examined: the site of a recognized international

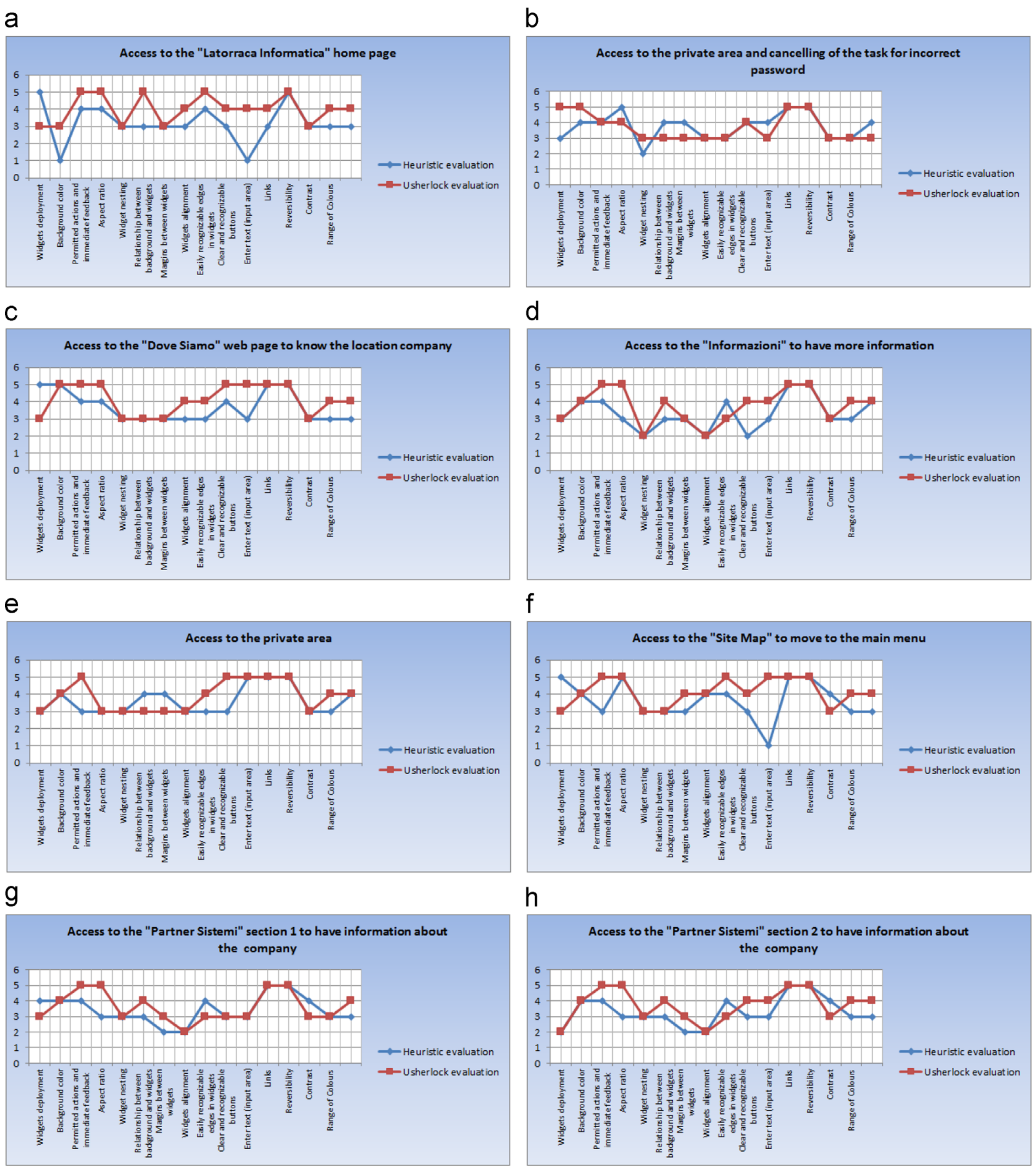


Fig. 19. The graphical results of a company web site.

organization dedicated to the issues of usability and accessibility of websites. The web pages analyzed are

- *http://www.w3.org/*
- *http://www.w3.org/standards/xml/*
- *http://www.w3.org/participate/*
- *http://open-stand.org/*
- *http://docs.webplatform.org/wiki/Main_Page*
- *http://www.w3.org/community/*
- *https://www.w3.org/community/wp-login.php?redirect_to=/community/groups/propose_cg*
- *http://www.w3.org/Consortium/membership*

Fig. 17 shows the comparative graphics for the considered web pages. Again, the title of each graphic indicates the task performed by participants during user tests.

The fourth site we analyzed is an Italian business site of information technology services (Fig. 18). The web pages analyzed are

- *http://latorracainformatica.it/*
- *http://latorracainformatica.it/area_riservata_11.html*
- *http://latorracainformatica.it/dove_siamo_10.html*
- *http://latorracainformatica.it/informazioni_12.html*
- *http://wsb.register.it/foman/form_mailing_action.jsp*
- *http://latorracainformatica.it/site_map.html http://latorracainformatica.sistemi.net/*
- *http://latorracainformatica.sistemi.net/pagina.asp?idlv=4&idDoc=8*

In Fig. 19 the comparative graphics are depicted. The title of each graphic indicates the task performed by participants.

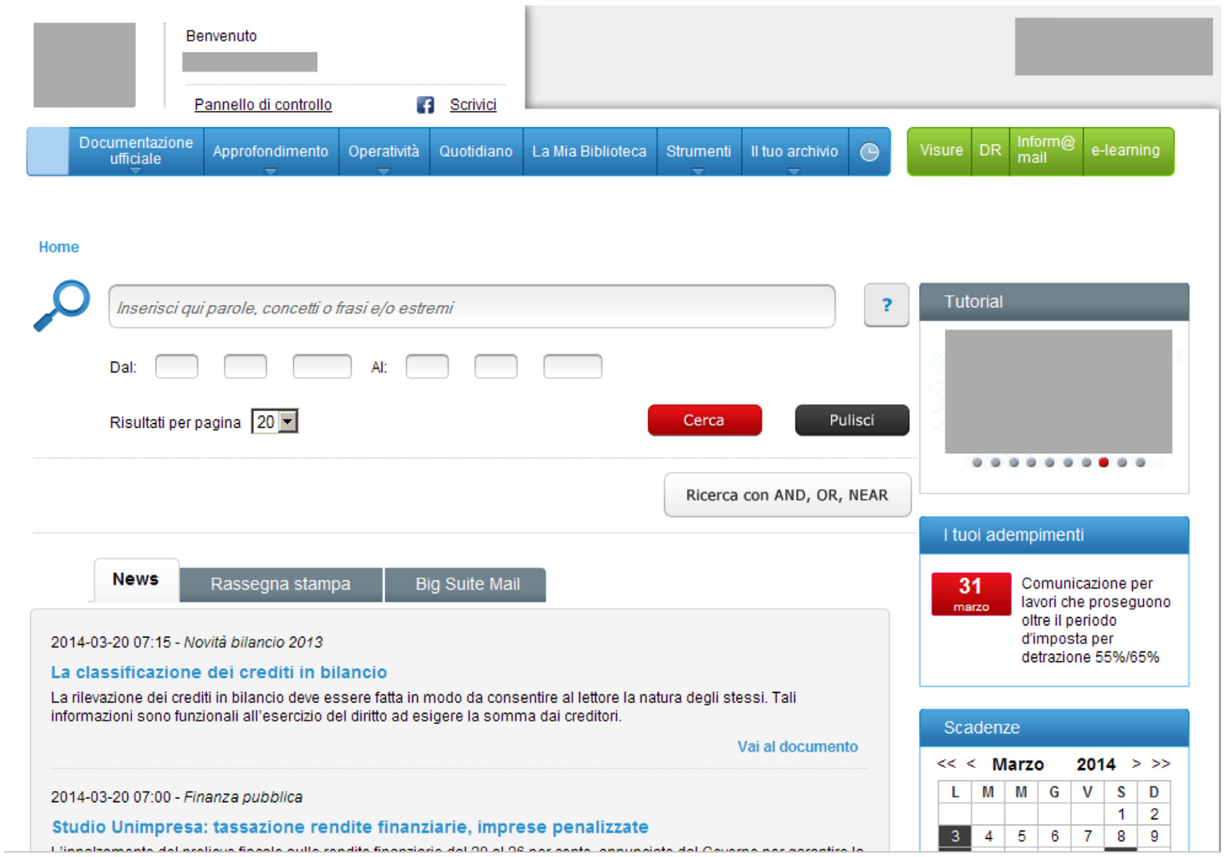
The last artefact we evaluated was the commercial visual database for professionals (see a screenshot in Fig. 20). The tasks considered are

- *enter the main interface*;
- *search information on tax deductions for first home owners*;
- *access the professional DB mail*;
- *access official documentation section*;
- *access further information section*;
- *access operational section*;
- *access tool section*;
- *access E-learning section.*

Fig. 21 shows the graphics of the average ratings achieved for the examined graphical user interface.

4.7. Data results

For each task, we analyzed 24 questionnaires resulting from the study and computed the mean values of the rates associated with each question. It is worth to note that, not surprisingly, a few outliers resulted from the experimental



Comunicazione per lavori che proseguono oltre il periodo d'imposta per detrazione 55%/65%

2014-03-20 07:15 - *Novità bilancio 2013*

La classificazione dei crediti in bilancio

La rilevazione dei crediti in bilancio deve essere fatta in modo da consentire al lettore la natura degli stessi. Tali informazioni sono funzionali all'esercizio del diritto ad esigere la somma dai creditori.

2014-03-20 07:00 - *Finanza pubblica*

Studio Unimpresa: tassazione rendite finanziarie, imprese penalizzate

<< < Marzo	2014 > >>					
L	M	M	G	V	S	D
					1	2
3	4	5	6	7	8	9

Fig. 20. A stand alone application.

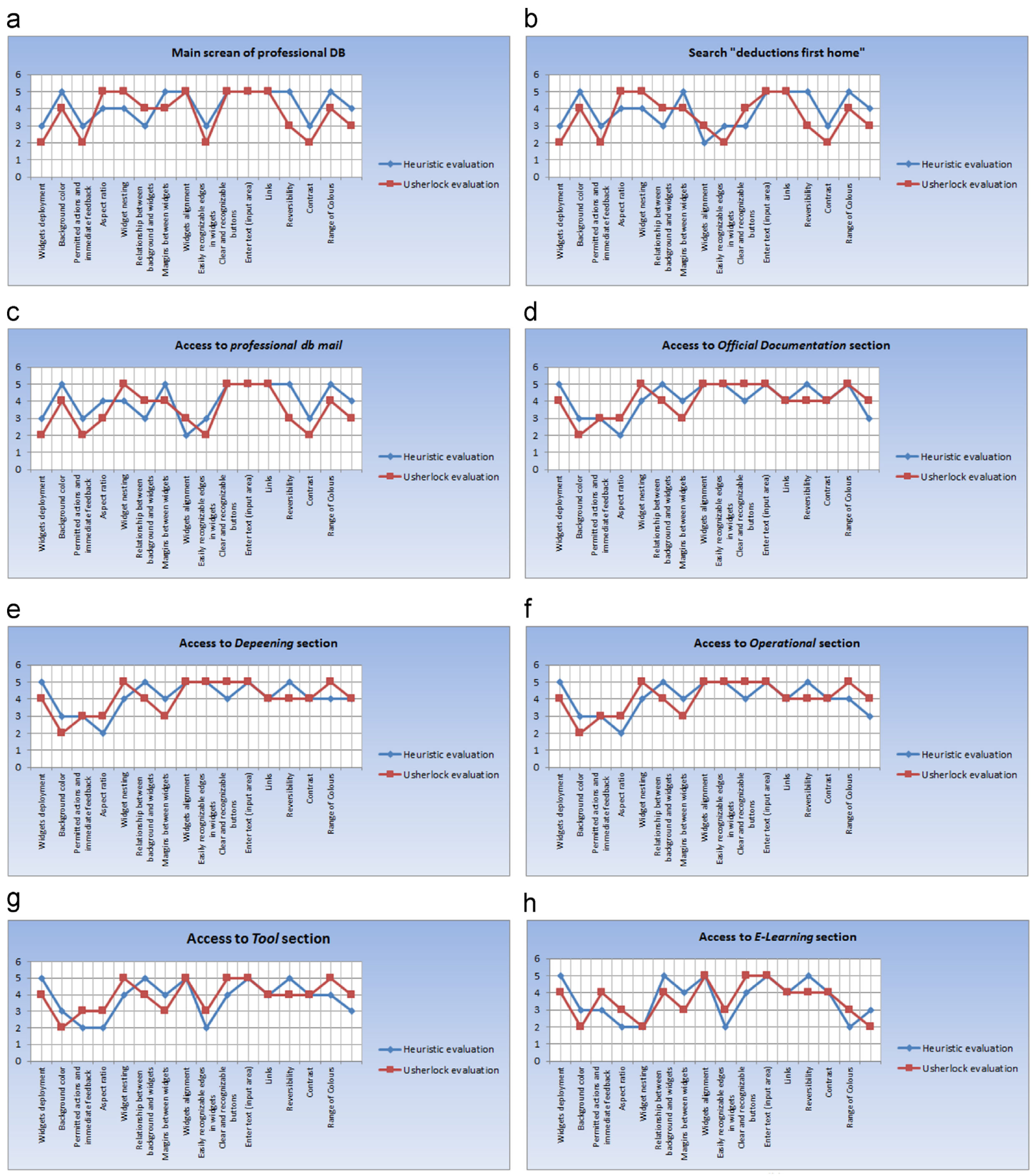


Fig. 21. The graphical results of a stand alone application.

scores datasets. Indeed, they may be due to the subjective nature of the evaluation and to the physical limitation of human sight (e.g., not perceiving lack of alignment between components) and therefore could be safely discarded before computing the mean values. As explained in the previous section, the graphics in Figs. 13,15,17,19, and 21 compare the average ratings collected during the experiment for the evaluated artefacts (four web sites and one stand-alone visual database). Such ratings were then used to compute effectiveness for each of the considered interfaces (eight interfaces per site) and perform a correlation analysis with respect to the evaluation method. Table 2 reports the

Table 2
Number of problems discovered.

	Number of problems discovered			*Effectiveness_USh (%)*	*Effectiveness_HE (%)*
	CG	HE	USh		
GoogleTask1	11	2	4	36	18
GoogleTask2	5	3	5	100	60
GoogleTask3	4	2	4	100	50
GoogleTask4	5	1	2	40	20
GoogleTask5	3	1	1	33	33
GoogleTask6	6	3	5	83	50
GoogleTask7	7	4	6	86	57
GoogleTask8	6	1	5	83	17
$\mu_{Effectiveness}$				**70**	**38**
MIUR_Task1	9	7	7	78	78
MIUR_Task2	9	7	8	89	78
MIUR_Task3	9	9	9	100	100
MIUR_Task4	9	7	9	100	78
MIUR_Task5	5	5	5	100	100
MIUR_Task6	6	4	6	100	67
MIUR_Task7	5	4	5	100	80
MIUR_Task8	6	6	6	100	100
$\mu_{Effectiveness}$				**96**	**85**
W3C_Task1	5	3	4	80	60
W3C _Task2	6	2	4	67	33
W3C _Task3	6	4	4	67	67
W3C _Task4	6	2	6	100	33
W3C _Task5	6	3	3	50	50
W3C _Task6	5	2	5	100	40
W3C _Task7	3	3	2	67	100
W3C _Task8	3	3	1	33	100
$\mu_{Effectiveness}$				**70**	**60**
CompanySite_Task1	8	5	6	75	63
CompanySite _Task2	8	6	5	63	75
CompanySite _Task3	6	4	5	83	67
CompanySite _Task4	9	8	6	100	89
CompanySite _Task5	9	5	8	89	100
CompanySite _Task6	4	4	3	75	100
CompanySite _Task7	6	4	3	100	67
CompanySite _Task8	8	4	6	75	50
$\mu_{Effectiveness}$				**82**	**71**
VisualDB_Task1	10	6	5	50	60
VisualDB_Task2	6	4	5	83	67
VisualDB_Task3	7	4	5	71	57
VisualDB_Task4	4	3	2	50	75
VisualDB_Task5	7	4	6	86	57
VisualDB_Task6	8	7	4	50	88
VisualDB_Task7	5	4	5	100	80
VisualDB_Task8	7	4	5	71	57
$\mu_{Effectiveness}$				**70**	**68**
EFF				**75**	**64**

number of problems discovered, respectively by the control group (expert evaluators), through USherlock and through heuristic evaluation. Data were collected separately for each of the performed tasks.

By comparing USherlock and heuristic evaluation averages for each query task, it is possible to notice that the values of effectiveness corresponding to USherlock are higher than the values corresponding to heuristic evaluation. The summary values corresponding to the mean values computed for each site also indicate that the USherlock method is more effective than the heuristic evaluation method. The purpose of our experiment has been to verify the validity of the above claims, showing that the improvements are unlikely to have occurred by chance. A one-tail t-test with a significance p-value < 0.05 has been applied on the collected results for the pair of methods we wanted to compare.

4.8. Hypotheses

We have formulated a pair of hypotheses, including the null hypothesis and an alternative, one-sided, hypothesis, as follows:

H0-1 (null hypothesis): The mean of effectiveness measure calculated using USh is not greater than the mean obtained using HE:
$EFF(\mathbf{USh}) \leq EFF(\mathbf{HE})$

Table 3
Results of the one-tail *t*-test.

Variable	Mean difference	95.00% confidence interval		*t*	*p*-Value
		Lower limit	Upper limit		
EFF_USherlock EFF_heuristic	0.106	0.018	0.193	2.443	0.019

H1-1 (alternative hypothesis): Ush is significantly more effective than HE:
$EFF(\mathbf{USh}) > EFF(\mathbf{HE})$

4.9. *Statistical analysis*

Before applying the *t*-test, we had to verify that the right conditions existed, namely that a normal distribution of data existed according to the Shapiro–Wilk test and equality of variances through Levene's test. Table 3 summarizes the results of the one-tail *t*-test, applied to the given data sets. It displays a *p*-value of 0.019, which is lower than the given threshold 0.05 and a positive $t = 2.43$. It implies that the null hypothesis can be rejected in favor of the alternative hypothesis and that a statistically significant improvement is achieved using USh rather than HE, in terms of effectiveness for the analyzed artefacts.

Overall, the analysis has confirmed the validity of the USherlock evaluation method giving statistical evidence to the claim that the method outperforms heuristic evaluation performed by practitioners with medium experience in usability evaluation. The percentage of problem detection with respect to an expert-based evaluation also suggests that the automatic evaluation method can be reliably adopted to reduce costs and time of testing activities.

5. Conclusion and further work

The goal of the presented research has been to validate the automatic usability evaluation method USherlock and give evidence to the degree of effectiveness achieved by that front-side approach during summative usability evaluation activities. Using USherlock we have examined a number of interfaces of four different web sites and one stand-alone application to calculate the average number of problems detected with respect to the number of real problems, as discovered by a group of expert evaluators. The results have been compared to those derived from an inspection evaluation method that involved students of a Human Computer Interaction and Software Usability academic course, playing the role of practitioners with medium experience in usability evaluation.

The outcomes of the empirical analysis show that an improvement in terms of effectiveness is indeed achieved using USherlock, with obvious advantages in terms of time and cost with respect to the canonical manual tests. Obviously, some remarkable subjective aspects of usability as the sense of satisfaction or the complexity of the path of an interaction are not measurable characteristics with automated methods. Nevertheless, the metrics evaluated using the proposed methodology can be a useful complement to standard techniques of evaluation.

In the future we are planning to further enhance the tool with the ability to test accessibility aspects of graphical user interfaces. This will require a new empirical analysis to evaluate its effectiveness. We will also investigate the inclusion of OCR systems and inference engines as solutions to improve the tool ability to recognize textual elements.

Acknowledgment

We wish to thank the anonymous reviewers for their comments and valuable suggestions to improve the quality of the paper. We are also grateful to the two expert evaluators and to the students who took part in the experiment.

References

[1] R. Cassino, M. Tucci, Automatic Usability Evaluation of GUI: A Front-Side Approach Using No Source Code Information (Lecture Notes in Information Systems and Organization), , 2013, 439–447.
[2] R. Cassino, M. Tucci, Developing usable web interfaces with the aid of automatic verification of their formal specification, J. Vis. Lang. Comput. 22 (2) (2011) 140–149.
[3] R. Cassino, M. Tucci, Usability evaluation of interactive visual applications: a quantitative approach, DMS'07, in: Proceedings of the Thirteenth International Conference on Distributed Multimedia Systems, 6–8 September 2007, San Francisco, Bay, USA, 2007.
[4] A. Fernandez, S. Abrahão, Empirical validation of a usability inspection method for model-driven web development, J. Syst. Softw. 86 (1) (2013) 161–186.
[5] A. Fernandez, E. Insfran, S. Abrahão, Usability evaluation methods for the web: a systematic mapping study, Inf. Softw. Technol. 53 (8) (2011) 789–817.
[6] W. Gray, M. Salzman, Damaged merchandise? A review of experiments that compare usability evaluation methods, Hum.–Comput. Interact. 13 (1998) 203–261.
[7] D.R. Hackner. A.M. Memon, Test case generator for GUITAR, in: Proceeding of the Companion of the 30th International Conference on Software Engineering, 2008, pp. 959–960.
[8] H. Hartson, T. Andre, R. Williges, Criteria for evaluating usability evaluation methods, Int. J. Hum.–Comput. Interact. 15 (1) (2003) 145–181.
[9] K. Hornbæk, Current practice in measuring usability: challenges to usability studies and research, Int. J. Hum.–Comput. Stud. 64 (2006) 79–102.
[10] E.T. Hvannberga, E.L. Lawb, M.K. Lárusdóttirc, Heuristic evaluation: comparing ways of finding and reporting usability problems, Interact. Comput. 19 (2) (2007) 225–240.
[11] M.Y. Ivory, M.A. Hearst, The state of the art in automating usability evaluation of user interfaces, ACM Comput. Surv. 33 (4) (2001) 470–516.
[12] Lu Lu, Automated GUI test case generation, in: Proceedings of the International Conference on Computer Science & Service System (CSSS), 2012, pp. 582–585.
[13] R. Mahajan, B. Shneiderman, Visual and textual consistency checking tools for graphical user interfaces, IEEE Trans. Softw. Eng. 23 (11) (1997) 722–735.

[14] A.M. Memon, Using reverse engineering for automated usability evaluation of Gui-based applications, in: A. Seffah, J. Vanderdonckt, M.C. Desmarais (Eds.), Human-Centered Software Engineering—Human–Computer Interaction Series, 2009, pp. 335–355.
[15] A.M. Memon, B.N. Nguyen, Advances in automated model-based system testing of software applications with a GUI front-end, Adv. Comput. 80 (2010) 121–162.
[16] A. Memon, A. Nagarajan, Q. Xie, Automating regression testing for evolving GUI software, J. Softw. Maint. Evol.: Res. Pract. 17 (1) (2005) 27–64.
[17] J. Nielsen, Usability Engineering, Academic Press, Boston, USA, 1993.
[18] J. Nielsen, Usability inspection methods, in: Proceeding of the CHI'94 Conference Companion on Human Factors in Computing Systems, pp. 413–441.
[19] P. Palanque, E. Barboni, C. Martinie, D. Navarre, M. Winckler, A model-based approach for supporting engineering usability evaluation of interaction techniques, in: Proceedings of the 3rd ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 2011, pp. 21–30.
[20] J.C. Silva, J. Creissac, J. Saraiva, GUI inspection from source code analysis, in: Proceeding of the Foundations and Techniques for Open Source Software Certification, Vol. 33, 2010.