Noname manuscript No.
(will be inserted by the editor)

Assessing the Quality of Mobile Graphical User Interfaces using Multi-objective Optimization

Makram Soui · Mabrouka Chouchane · Mohamed Wiem Mkaouer · Marouane Kessentini · Khaled Ghedira

the date of receipt and acceptance should be inserted later

Abstract Aesthetic defects are a violation of quality attributes that are symp-toms of bad interface design programming decisions. They lead to deteriorating the perceived usability of mobile user interfaces and negatively impact the Users eXperience (UX) with the mobile app. Most existing studies relied on a subjective evaluation of aesthetic defects depending on end-users feedback, which makes the manual evaluation of mobile user interfaces human-centric, time-consuming, and error-prone. Therefore, recent studies have dedicated their effort to focus on the definition of mathematical formulas that each targets a specific structural quality of the interface. As the UX is tightly dependent on the user profile, the combi-nation and calibration of quality attributes, formulas, and users characteristics, when defining a defect, is not straightforward. In this context, we propose a fully automated framework which combines literature quality attributes with the users profile to identify aesthetic defects of MUI. More precisely, we consider the mobile user interface evaluation as a multi-objective optimization problem where the goal is to maximize the number of detected violations while minimizing the detection complexity of detection rules and enhancing the interfaces overall quality in means

M. Soui
College of Computing and Informatics Saudi Electronic University, Saudi Arabia
E-mail: m.soui@seu.edu.sa

Mabrouka Chouchane
School of computer science of Manouba, Tunisia
E-mail: chouchane.mabrouka@gmail.com

Mohamed Wiem Mkaouer
Rochester Institute of Technology
E-mail: mwmvse@rit.edu

Marouane Kessentini
University of Michigan
E-mail: marouane@umich.edu

Khaled Ghedira
Honoris United Universities
E-mail: khaled.ghedira@universitecentrale.tn

of guidance and coherence coverage. We conducted a comparative study of several evolutionary algorithms in terms of accurately identifying aesthetic defects. We reported their performance in solving the proposed search-based multi-objective optimization problem. The results confirm the efficiency of the Indicator-Based Evolutionary Algorithm (IBEA) in terms of assessing the developers in detecting typical defects and also in generating the most accurate detection rules.

Keywords Mobile user interface · Aesthetic quality · Evaluation

1 Introduction

The mobile environment has been advancing towards being the leading medium for the worlds global network [97]. As a consequence, the tremendous growth of mobile applications offered to users a wide variety of applications (apps) with similar functionalities.This huge evolution of mobile apps induce to increasing attention to mobile user interfaces as the graphical part of the mobile device that allows the user to interact with the devices apps, features, content and functions. The quality of Mobile User Interfaces (MUIs) is a key factor in the mobile application effectiveness and the user satisfaction. In addition, according to [96], the user interface represents an important part of the application.So, assessing the MUI helps to evaluate the interaction and usability of the overall system. Furthermore, user interface level represents 50% of software code [64,70] which proves the importance of this level in the correctness and the effectiveness of the mobile application. [42] reported that MUI-related defects have a significant impact on the end users of the mobile applications. He has shown that 60% of defects can be traced to code in the Graphical User Interface (GUI), and 65% of GUI defects resulted in a loss of functionality. Therefore, evaluating a MUI is a very important phase in the development to decrease the maintenance cost of mobiles applications. In fact, detecting the aesthetic defects help the evaluator to quickly enhance the quality of MUIs as the aesthetic represents the beauty or the pleasing appearance of user interfaces of a mobile app that underlines a vital sub-characteristic of the overall usability [99]. Recently, reviews on the state of the art about evaluation methods for GUI have been tackled by several researchers [34,94,104]. However, there is no consensus on how mobile user interface should be assessed.

Various studies [25, 51, 19, 36, 86] have focused on the optimization of User Interface (UI) quality mainly by identifying aesthetic design defects. These defects are violations of the structural design principles, defined in the literature [54, 30, 12, 83, 11]. These approaches define each defect through a combination of structural metrics, extracted from UIs, and search for their appropriate threshold that, combined, become problematic. For instance, a defect like *overloaded MUI* may be identified by a large number of interactive objects in a UI, with respect to its alignment points, or by a high value of UI complexity. Unfortunately, the detection and correction of aesthetic design defects in non-trivial software systems can be very challenging: The generation of detection rules for these defects, using structural metrics, tends to be subjective and user-specific. For instance, there

are multiple ways to correlate *overloaded UI* with structural metrics. Also, the thresholds for these metrics differ from one user to another. For the example of *overloaded UI*, what is considered to be an abnormal number of interactive objects differs from one user to another, as some users are more familiar with loaded interfaces and others tend to opt for a more simplistic design [65]. The optimal way to tackle this problem, is to generate all possible combinations of metrics, with their corresponding possible threshold values, for each defect, and evaluate it for each type of user. As the number of possible combinations of metrics, correlated with possible user profiles, tends to be very large, the exhaustive approaches tend to be impractical. Based on these considerations, we suggest solving the detection of defects process as a multi-objective optimization problem, where we search for the trade-off of generating detection rules while maximizing their correctness and minimizing their complexity.

In the area of HCI, several studies analyzed the quality of GUIs using the manual definition of what is considered to be a bad design decision, the detection of such patterns use the declarative rule definitions [88,57,58]. These rules are manually defined to identify defects using combinations of quantitative metrics and context criteria (user experience, user motivation, age, etc.). These evaluation metrics are based on a set of interface attributes (number of interface components, the number of the alignment point, the number of colors, etc.). More specifically, in an exhaustive scenario, the number of possible metrics to deploy and the number of context criteria used to manually design rules tend to be vast. Moreover, a calibration effort is needed for finding the best threshold for evaluation metrics and/or context criteria that constitute each rule. In fact, it is very difficult to generalize these rules for all mobile interfaces when taking into account heterogeneous user profiles. Another interesting observation is that translating defects into rules is not straightforward because there is no consensus on what can be seen as a defect on the GUI according to calculated metrics without a manual validation from the user. Moreover, the same mobile user interface attributes could be associated with multi-evaluation metrics. When consensus exists, the same evaluation metrics could be related to multi-defect types. In addition, the majority of existing work proposes to manually define the detection rules [88,57,26]. This task is complex, time-consuming and subjective.

In addition, it was a challenge for developers to provide users with a better user experience. The importance of involving user profiles in the design and evaluation of Graphical User Interfaces (GUIs) has been advocated in the recent studies [35, 59,62]. However, and to the best of our knowledge, no previous study has explored the evaluation of GUIs quality in function of users characteristics. In this context, our goal is to assist developers when designing Mobile User Interfaces (MUI) by evaluating their quality based on the nature of potential users of their apps. Most of the existing tools and frameworks are mainly testing the apps from a functional perspective, along with helping developers in automating code- reviewing, debugging, and code design smells. Only few studies assess the quality of the apps GUI design, while taking into account the users profile, yet GUIs are critical in influencing the users perception of the app and its functionality [56,20,53]. Therefore, the design of an adequate GUI enhances the users satisfaction regarding the high

number of competing applications that are continuously growing in several app stores and mobile apps markets.
To our best knowledge, there are no multi-objective evaluation solutions dedicated to evaluating the quality of GUIs within mobile apps, and taking into account user characteristics. For this purpose, this research focuses on studying the popular metrics, generally used in evaluating GUIs, and adapted them to the context of mobile apps based on the following main properties:
Structural design metrics Mobile GUIs are a family of a larger category of event-driven graphical interfaces. Consequently, just like any software system, they share design characteristics that can be measured and evaluated through the static analysis of the GUIs source code. In other terms, several metrics such as complexity, symmetry and repartition of components, once combined, can be used as indicators of good or bad design decisions when placing components within an interface.
Focused interfaces The regular applications rely on training and documentation to provide guidance for users to use them. However, mobile apps, are expected to be more intuitive. In fact, these apps are used in smaller sized input mediums that are still able to accept more diverse user gestures and commands. More concretely, the GUI has not only to adequately respond to most, if not all, gestures that may look evident to common users, but also it is expected to be intuitive while accommodating a vast variety of potential users and to close the gap between generations. To this end, typical context criteria, such as age, experience, and motivation etc. are used to classify users into families and guide the app architect into deciphering what is expected to be in the GUI from the user's perspective.
Since the end-users are the final judges of UI design choices, it is impractical to coin UI design defects without taking into account their profile as a part of the definition. Due to the diversity of target app audience, and the large number of design properties of GUI, there is a difficulty to express the defects of user interfaces. Consequently, our work focuses on how to combine these two dimensions to model the symptoms of GUI quality deterioration.

Thus, one of our contributions is the automation of the generation of rules instead of the manual process. Various types of structural aesthetics metrics have been adopted to evaluate the quality of GUIs. Using our approach as a development support tool, developers can take as input the user's feedback on their UIs, along with their characteristics, to automatically generate rules for what users have considered to be problematic for in the UI design. To build on existing studies, we gathered the structural aesthetics metrics to cover the visual com- plexity and the visual aesthetics of the GUI [95, 84, 66, 12]. As found out by the experimental study done by Riegler and Holzmann [77], any variation of elements sizes, text fonts, or colors will increase the complexity of the MUI. As these parameters represent some visual aspects of the MUI, we will categorize them under what we call a visual complexity to avoid using the nomination of complexity that undertakes another attributes of the GUIs. This research aims to address these issues by proposing a GUI evaluation approach that takes as input from one dimension a set of quality metrics, context criteria and defect types and from another dimension the users feedback of their usage to a given set of GUIs and it will automatically generate

detection rules of what users saw as problems in the GUIs. Due to the stochastic nature of the rules generation, we considered it as an optimization problem. Consequently, a comparative study of several search-based algorithms was conducted. The validation of this work shows that we were able to automatically generate rules that can replicate the defects identified by the users. These rules can be used to potentially evaluate GUIs in the upcoming app releases and prevent the leaking of any GUI defect.

To summarize, the main explored problem in this study centers around the automation of the MUI aesthetics quality evaluation process. Precisely, we tack-led the relationship between the MUI structural design and user's perception. We considered five user's profile demographics attributes to find a trade-off between how each type of users perceive the aesthetics quality of the MUI. We mainly contribute to the research community and to the area of HCI with a set of evaluation rules that considers different types of addressed final users. Our proposed approach differs widely from the previous proposed approaches, as we consider the formulation of the evaluation rules as an optimization problem, where we automatically and genetically produce a set of heuristics rules that efficiently map between the user profile, metric and defect. In order to assure the maximum rules coverage, we compared four genetic algorithms NSGA-II, MOAD,SPEA2 and IBEA. The Users data has been collected through an experimentation where 200 MUIs have been evaluated by 20 students. To perform our approach we, proceeded in the following way:

- We studied the state-of-art metrics, widely used in the HCI, to evaluate UIs and we adapted them to the context of GUIs. We implemented a plug-in that calculates these metrics through the static analysis of the GUIs source code called PLAIN.
- We automatically combined the structural metrics with the user context criteria to generate detection rules for UI defects.Due to its stochastic nature, we defined rules generations as a multi-objective optimization problem.Our objective is to generate rules that adequately represent the user's perspective of a bad UI while minimizing the number of metrics involved in this definition to reduce the detection complexity.
- We conducted a comparative study between several search-based algorithms, and we reported their performance in terms of convergence towards an optimal state i. e., identifying the defects previously reported by the users. The algorithms performance was tested from a functional perspective i.e., their ability to generate correct, yet simpler rules, and from an evolutionary perspective i.e., the ability of search heuristics to converge towards near-optimal solutions.
- A quantitative and qualitative evaluation of this approach on several real-world mobile apps has shown its efficacy in detecting defects in particular and enhancing the GUI quality in general.

The results confirm the efficiency of the Indicator-Based Evolutionary Algorithm in terms of assessing the developers in detecting typical defects and also in generating the most accurate detection rules.

The rest of the paper is organized as follows: Section 2 presents the necessary background of GUI evaluation and demonstrates the problems tackled in this research with regard to the description of context modeling, a set of quality metrics and a set of defects. Section 3 describes the definition of rules generation as an optimization problem. Results discussion of the comparative study is elaborated in section 4. The threats to validity of our study is reminded in section 5. Section 6 details existing studies in evaluating quality of UIs. Finally, Section 7 gathers our conclusions along with some pointers to future work.

2 Background: Graphical User Interface

2.1 Evaluation principle

In the literature, several authors [78,10] define the interface evaluation as the software unity which improves its interaction with a user by the construction of users model based on its crossed interactions with this user. The mobile user interfaces must reach the users needs more easily, faster and with a higher level of satisfaction. Such evaluation is very subjective to the users overall usage. According to [3], usability is defined as the effectiveness, efficiency, and satisfaction with which a set of users can achieve a set of tasks in a defined environment. According to [28,44] user-centered evaluation aims to verify the quality of user interface, detect defects and propose recommendation. More precisely, every evaluation includes identifying and predicting the obstacles facing the users that are preventing them from easily reaching out to system functionalities. For the evaluators, the construction of the model is guided by gathering as much information as possible about the users characteristics, in order to create a context of use, based on these characteristics; that is to say, the identification of problematic properties that deteriorate the interface quality does not only depend on structural properties but also on target users characteristics. These properties are the utility and usability [21,24, 32]. To summarize, GUI diagnosis consists in ensuring that the system provides to a user with relevant information via an interface on two levels:

- At the content and services level: ensuring that services proposed by the system correspond to user requirements. This evaluation is out of the scope of our work.
- At the user interface level: it verifies how successful the interfaces are (style of display, components distribution, etc.) in coping with user preferences.

The main objective of this paper is to analyze and automate such a non-functional evaluation.

2.2 Context modeling

Several studies emphasize the importance of incorporating the users in the loop in several engineering tasks such as modeling [73], testing [46] and bug triage [103]. However, taking into account the contexts characteristics is a support for GUI evaluation. For instance, a GUI can be considered as plastic or multi-platform

graphical user interface and be adapted to the user, the platform, and the environment by respecting its usability [93,38,39]. The concept of context was defined by [29] as follows: Context is any information that can be used to characterize the situation of an entity. The entity can be a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves. For this paper, we have retained the definition presented in [93]. This definition of the context based on three dimensions: user, platform, and environment. When evaluating a GUI, each one of the contexts dimension can be characterized by several elements. Thus, the user could be characterized by:

- Age.
- Interest: is a feeling of a user whose attention, concern, or curiosity is particularly engaged when interacting with the GUI, etc. [31].
- User experience: which is computer skills of the user, etc., [80,100,55,92,81].
- Motivation: which is a factoring motivated to act or accomplish that stimulate desire and energy in user or having a strong reason to be continually interested and to make an effort to achieve a goal.

The platform could be divided into two sub-parts (hardware and software):

- The hardware: which is all platforms specific technical aspects including all the interaction possibilities, screen size, memory, etc. [98,90,14,91].
- Software: which is the set of applications installed on the platform including an operating system, virtual machine,. [81,90,75].

The environment could be characterized by:

- Time: the year, month, day, morning, evening, public holiday, etc. [52,15,43, 74].
- Geographical element: GPS coordinates or the type of space like a station or a store,... [31,71,43].
- Physical environment or environment condition like luminosity, noise, weather, ... [80,55,92,81,75]. For example, the luminosity which is a measurement of brightness or light can involve a change at the level of the used colors. It can be changed suddenly when the user passes from the shade to the sun such as the luminosity of the screen, the back-lighting of the keyboard.

In our experiments, we use just five context criteria which are: age, interest, motivation, education level and user experience.

2.3 Aesthetics Defects of GUI

In the literature, we find a large list of aesthetic defects of user interface. In our study, we focus on the following eight defects presented in table 1 and proposed by [47]:

Table 1 List of Aesthetic Defects

Aesthetics Defects	**Description**
Incorrect layout of widgets	It is related to the incorrect arrangement of MUI components. [54]
Overloaded MUI	It is a bad density of MUI. In other words, users find the mobile interface too dense and so difficult to read. [30]
Complicated MUI	It is related to the MUI that includes too many widgets and features which cannot meet the users' needs.[12]
Incorrect data presentation	It is the incorrect extraction of information and their display on the mobile screen.[83]
Lack of Cohesion in MUI	It is the lack of the interrelatedness of MUI components.[11]
Difficult navigation	It is the of lack descriptive labels that can be used to define the additional information.[4]
Ineffective appearance of widgets	It occurs when MUI widgets follow an unexpected layout.[30]
Imbalanced MUI	It is an unequal distribution of the quantity of interactive objects of a MUI.

2.4 Adaptation of evaluation metrics for a mobile user interface

Several metrics have been used in HCI for the purpose of evaluating GUIs. In this work, we use a set of evaluation metrics that were previously validated using several ergonomic criteria proposed by [87]. In fact, these structural metrics are adapted to evaluate the structural aspect of GUI. In this study, we used metrics proposed by [87] that can be classified into two criteria: (1) guidance (2) coherence.

– **Guidance:** User guidance refers to the means available to advise, orient, inform, interact, and guide the users throughout their interaction with the computer (message, alarm, label, etc.). This criterion is subdivided into four metrics: regularity, composition, sorting, and complexity.

Regularity

The regularity metric allows a GUI to have a consistent spacing between all its components while minimizing the number of rows and columns of these interfaces (alignment points). The purpose of this metric is to arrange the GUI components in order to preserve their structure. This metric has an influence on user criteria such as age, motivation, etc. In addition, the regularity of the GUI offers users the opportunity to satisfy their needs more easily.
In the context of mobile applications, the interfaces that have a high level of regularity are the most relevant. We use the following formula to measure this metric.

$$RM = 1 - (\frac{N_{av} + N_{ah} + N_{sp}}{3n}) \in [0, 1] \quad (1)$$

Where:
N_{av}:the numbers of vertical alignment points (number of rows).

N_{ah}:the numbers of horizontal alignment points (number of columns).
N_{sp}:the number of distinct distances between column and row starting points.
n:the number of the component of the mobile user interface. An example of this metric is shown in figure 1.

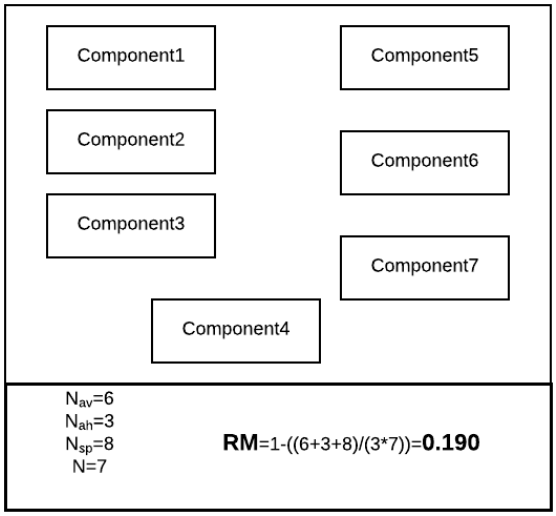

Fig. 1 An example of regularity metric calculation.

Composition

This metric serves to increase the visual clarity of the GUI by the meaningful arrangement of the interface components.The primary objective of this metric is the visual and semantic grouping of these components that are related in the same border (the line, the color, the shape, etc.).These metric counts the number (number) of objects that have a clear boundary by the group. Generally, GUI composition helps younger or older users with low computer skills to interact with the system. In fact, this kind of user prefers GUI with the high composition. To measure this metric, we use the following formula:

$$COM = 1 - (\frac{G + UG}{2n}) \in [0, 1] \tag{2}$$

Where:
G:the number of groups with clear boundary by line, background, color, or space.
UG:the number of ungrouped objects
n:the number of the component of the mobile user interface. An example of this metric is shown in figure 2.

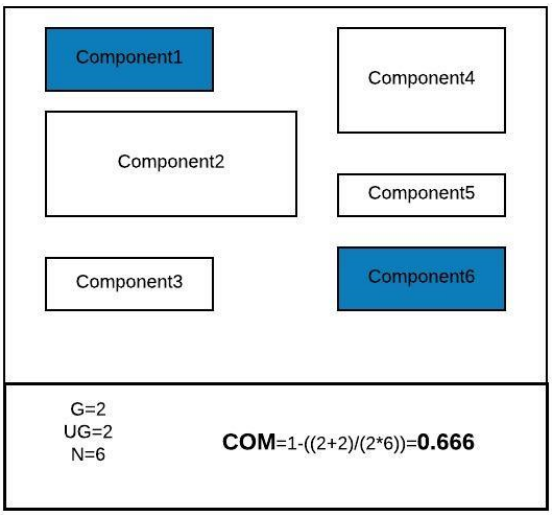

Fig. 2 An example of composition metric calculation.

Sorting

It ranks the GUI components according to the eye movement that moves sequentially from a dark to a lighter area, from big to little objects, etc. However, it arranges the component in a logical and sequential ordering that refers to the users needs. In fact, it helps the user throughout their interaction with the mobile user interfaces by offering to the elderly users an ordered interface with a high level of sorting. We propose the following formula to calculate this metric:

$$SM = 1 - (\frac{(\sum_{j=UL,UR,LL,LR}(q_j \sum_{i=1}^{n} N_{i,j}))}{4n}) \in [0,1] \quad (3)$$

Where:
UL: upper-left
UR: upper-right
LL: lower-left
LR: lower-right
$N_{i,j}$: is the number of object on the quadrant j.
Each quadrant is given a weighting in q_j. So,$q_{UL=}$=4,q_{UR}=3,q_{LL}=2,q_{LR}=1.
An example of this metric is shown in figure 3.

Complexity

Complexity metric aims to keep up not only an optimal number of interactive objects in GUI but also a minimal number of alignment points. It offers to the user only the relevant information that satisfies their needs and expectation. In fact, novice users usually prefer an interface with a low level of complexity while users having higher education level prefer MUI with a high level of complexity. In our work, we calculate the complexity metric as follows:

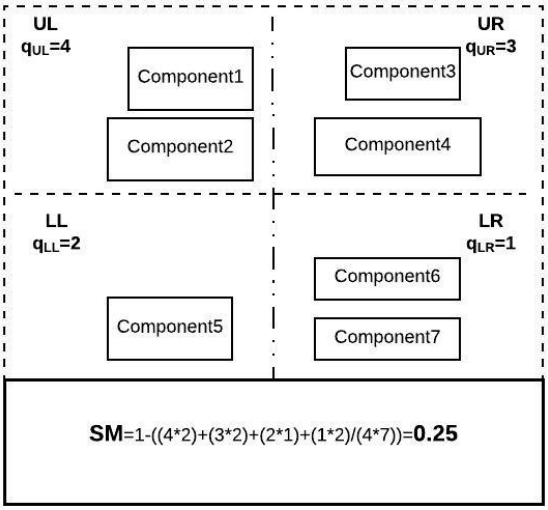

Fig. 3 An example of sorting metric calculation.

$$CM = (\frac{N_{vap} + N_{hap}}{2n}) \in [0, 1] \tag{4}$$

Where:
N_{vap}= number of vertical alignment points.
N_{hap}= number of horizontal alignment points.
n = number of object on the mobile user interface.
An example of this metric is shown in figure 4.

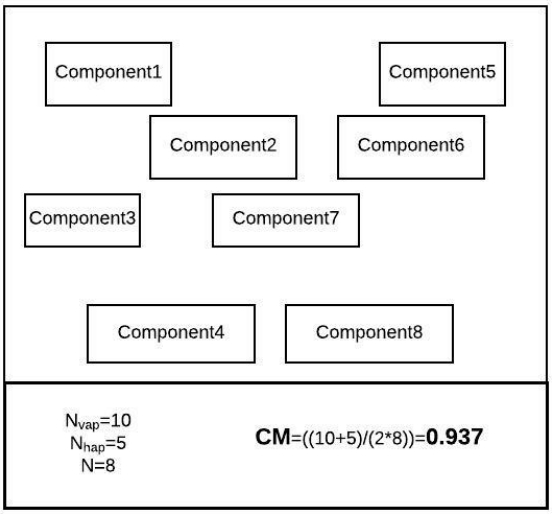

Fig. 4 An example of complexity metric calculation.

– **Coherence** It supplies how good the interaction between users and mobile user interfaces is, and secures the efficient use of the MUI. This criterion is subdivided into four metrics: integrality, density, repartition, and symmetry.

Integrality

Integrality metrics arranges all GUI components in order to seem to be a one piece. In fact, the screen size of the smartphone is not the same as of a computer that is why it is necessary to adapt the information quantities and form of information. it is the extent to which the components are correlated with the size, number of space between groups and margins. A good integrality is obtained by using the optimum number of size component (minimize the uses of different sizes in the mobile interface) and leaving less space between objects. When the level of integrality increases, the mobile interface is not centered as well. The formula of This metric is given as follow:

$$IM = 1 - (0.5 \left[\frac{|N_{size} - 1|}{n} + \frac{|a_{sc} + \sum_i^n a_i|}{2a_{MUI}} \right]) \in [0, 1] \quad (5)$$

Where:
N_{size} = the number of various sizes belong into used objects.
n = the number of objects.
a_{MUI}= the area of the mobile interface.
a_{sc} = the area of the screen.
a_i = the area of the interactive object i.
An example of this metric is shown in figure 5.

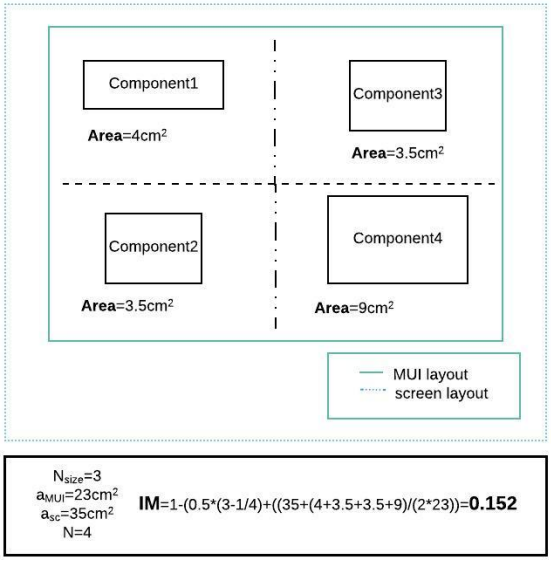

Fig. 5 An example of integrality metric calculation.

Density

This metric aims to minimize the number of component in an MUI. The level of this metric depends on the motivation, experience, interest of users and the type of target platform. In fact, a user with a low motivation prefers an MUI with a low-density level. We can calculate it as follow:

$$DM = 0.5 \left| \frac{\sum_i^n a_i}{a_{MUI}} + \frac{a_{MUI}}{a_{sc}} \right| \in [0, 1] \quad (6)$$

Where:
a_i= area of the interactive object i.
a_{sc}= area of the screen of interactive platform.
n= number of interactive object.
a_{MUI}= the area of the mobile user interface.
An example of this metric is shown in figure 6.

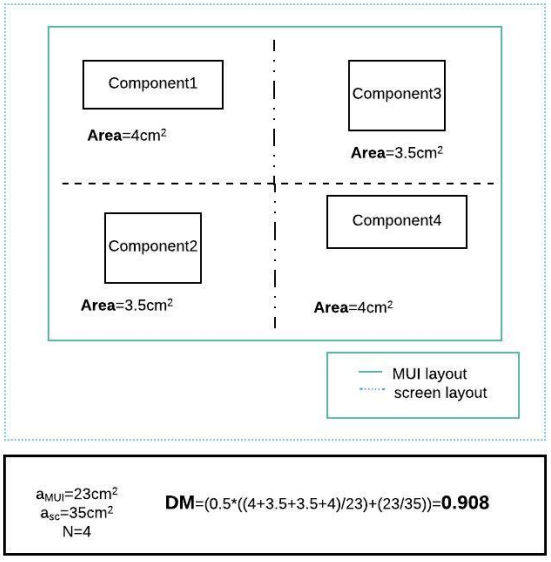

Fig. 6 An example of density metric calculation.

Repartition

This metric aims to provide an equal arrangement of all component of GUI among the four quadrants (upper left, upper-right, lower-left, lower-right). It calculates the numbers of different ways that objects can be organized in the four quadrants of GUI, compared to an optimal distribution. In fact, for a GUI with N components, there are n! different ways to arrange them. However, the optimal distribution is obtained when the N objects are uniformly distributed

in the four quadrants of the GUI. For example, a GUI should propose an optimal distribution for novice users in order to help them to navigate through it. The formula of this metric (RM) is given as follow:

$$RM = \frac{(\frac{n}{4}!)^4}{n_{UL}!n_{UR}!n_{LL}!n_{LR}!} \in [0,1] \quad (7)$$

Where:
n: is the number of object on the mobile user interface.
n_{UL}: is the number of object on the upper-left.
n_{UR}: is the number of object on the upper-right.
n_{LL}: is the number of object on the lower-left.
n_{LR}: is the number of object on the lower-right.
An example of this metric is shown in figure 7.

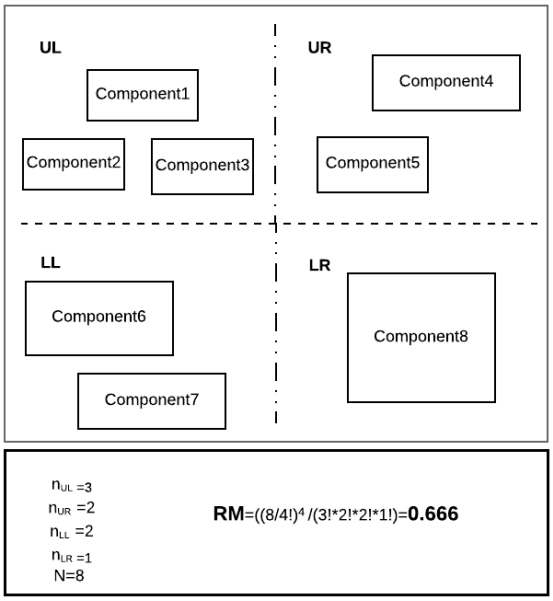

Fig. 7 An example of repartition metric calculation.

Symmetry

Symmetry aims to distribute uniformly the number of the component on the right and the left columns of a GUI while duplicating it on the left, right, and radical of GUI center-line. It avoids the imbalance in the different part of GUI. For example, this metric adjusts the GUI for users with low motivation in order to stimulate their interests. In our work, we use the formula proposed by [66].

$$SYM = -\frac{|SYM_{vertical}| + |SYM_{horizontal}| + |SYM_{radial}|}{3} \in [0,1] \quad (8)$$

$Sym_{vertical}, Sym_{horizontal}, Sym_{radial}$ = The vertical , horizontal and radial symmetries with :

$$\mathrm{Sym}_{\mathrm{vertical}} = \frac{\begin{array}{l}|X'_{UL} - X'_{UR}| + |X'_{LL} - X'_{LR}| + |Y'_{UL} - Y'_{UR}| + \\ |Y'_{LL} - Y'_{LR}| + |H'_{UL} - X'_{UR}| + |H'_{LL} - H'_{LR}| + \\ |B'_{UL} - B'_{UR}| + |B'_{LL} - B'_{LR}| + |\Theta'_{UL} - \Theta'_{UR}| + \\ |\Theta'_{LL} - \Theta'_{LR}| + |R'_{UL} - R'_{UR}| + |R'_{LL} - R'_{LR}|\end{array}}{12}$$

$$\mathrm{Sym}_{\mathrm{horizontal}} = \frac{\begin{array}{l}|X'_{UL} - X'_{UR}| + |X'_{LL} - X'_{LR}| + |Y'_{UL} - Y'_{UR}| + \\ |Y'_{LL} - Y'_{LR}| + |H'_{UL} - X'_{UR}| + |H'_{LL} - H'_{LR}| + \\ |B'_{UL} - B'_{UR}| + |B'_{LL} - B'_{LR}| + |\Theta'_{UL} - \Theta'_{UR}| + \\ |\Theta'_{LL} - \Theta'_{LR}| + |R'_{UL} - R'_{UR}| + |R'_{LL} - R'_{LR}|\end{array}}{12}$$

$$\mathrm{Sym}_{\mathrm{radial}} = \frac{\begin{array}{l}|X'_{UL} - X'_{UR}| + |X'_{LL} - X'_{LR}| + |Y'_{UL} - Y'_{UR}| + \\ |Y'_{LL} - Y'_{LR}| + |H'_{UL} - X'_{UR}| + |H'_{LL} - H'_{LR}| + \\ |B'_{UL} - B'_{UR}| + |B'_{LL} - B'_{LR}| + |\Theta'_{UL} - \Theta'_{UR}| + \\ |\Theta'_{LL} - \Theta'_{LR}| + |R'_{UL} - R'_{UR}| + |R'_{LL} - R'_{LR}|\end{array}}{12}$$

$X_j = \sum_{n=i}^{n_j} |X_{ij} - X_c|$

$Y_j = \sum_{n=i}^{n_j} |Y_{ij} - Y_c|$

$H_j = \sum_{n=i}^{n_j} h_{ij}$

$B_j = \sum_{n=i}^{n_j} b_{ij}$

$\Theta_j = \sum_{n=i}^{n_j} |\frac{Y_{ij} - Y_c}{X_{ij} - X_c}|$

$R_j = \sum_{n=i}^{n_j} \sqrt{(X_{ij} - X_c)^2 + (Y_{ij} - Y_c)^2}$

with j ∈ {UL,UR,LL,LR} Where UL, UR, LL, and LR are respectively: Upper-Left, Upper-Right, Lower-Left, and Lower-Right.

X_j: is the total x-distance of quadrant j.

Y_j: is the total y-distance.

H_j: is the total height.

B_j: is the total width.

θ_j: is the total angle.

R_j: is the total distance.

$(x_{ij}; y_{ij})$: the coordinates of the centres of object i on quadrant j.

$(x_c; y_c)$: the coordinates of the frame.

b_{ij}: the width of the object.

h_{ij}: the height of the object.

n_{ij}: the total number of objects on the quadrant.An example of this metric is shown in figure 8.

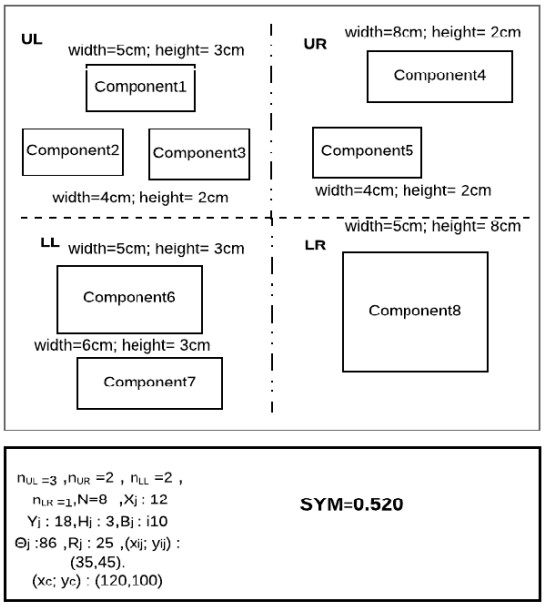

Fig. 8 An example of symmetry metric calculation.

3 Approach

We propose an automatic approach that uses the metrics mentioned above to define possible GUI bad decisions, based on any knowledge from previous GUIs evaluations.
The proposed approach is called AQUA (Assessing the quality of Mobile User Interfaces).As depicted in Figure 9, our approach has three phases:
1) collection of defects examples (A1)
2) the extraction of evaluation rules (A2)
3) the evaluation of GUI quality through defects detection (A3).

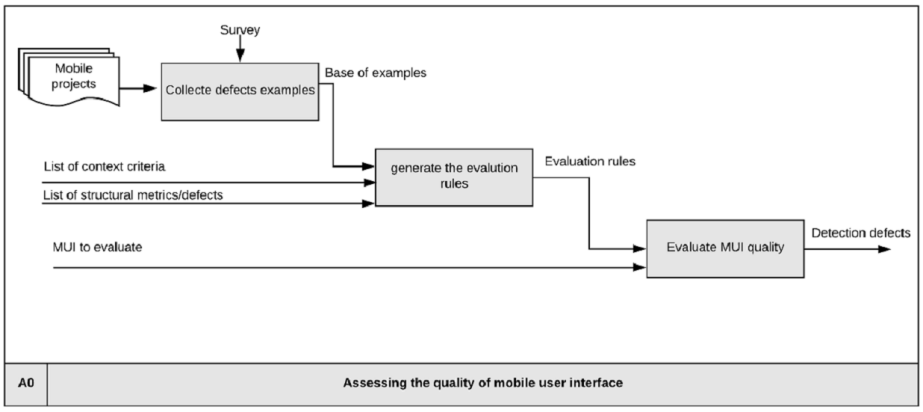

Fig. 9 Approach overview.

3.1 Data Collection

3.1.1 Extraction of defect examples

This phase consists of the construction of a base of defects examples that will guide the extraction of evaluation rules. It includes two steps: the first one is the collection of users opinion. The base of examples can be seen as a pair of the users' profile, which is defined by several properties that characterize the user and its context of use, and the second step captures his/her opinion about any problems identified in the GUIs. The subjects were invited to fill a questionnaire that aims to evaluate user interfaces of the studied projects. This questionnaire was divided into two parts:

- User profile: Contains the profile of the suggested user.
- User evaluation: Contains the answer of the user to the demanded questions after testing the project.

In this context, the subjects were first asked to fill out the first part of the questionnaire that contains seven questions. Then, we collect the profile of all the users. Moreover, we propose for each user the appropriate interface according to his profile. We used five context criteria (age, motivation, education level, user experience and interest) that can take three values (Low, Medium, and High). However, the subjects have different values for the criterion age. So, we classify the user according to their age into three intervals: the first is in [18,30], the second is in [30,55] and the third is in [55,80], and we accord the values low, medium and high respectively. After answering the first part, the subjects should test their appropriate interface and evaluate in order to detect the quality defects such as overloaded of GUI, incorrect layout of widgets, complicated GUI, interface lack of cohesion, imbalance GUI. The subjects are asked to express their satisfaction after testing their appropriate interface. So, they are invited to select for each question one of the possibilities: "Yes", "No", or "Maybe" (if not sure).Their evaluation results are reviewed by two observers and an expert in mobile user interface design.This questionnaire is distributed among the subjects in a pre-experimental

setup. Since we collect the result of evaluation from the questionnaire, we organize it into a survey(trace) to validate the evaluation rules generated from our approach. Figure 10 shows an example of an evaluated interface labeled preferences extracted from a mobile app called lirbi, and the revealed problems by the evaluators. As explained in the survey, this interface has been identified as difficult navigation defect (low composition) by a user with medium motivation while the same interface has not been seen defective by another user with higher motivation.

	Age	User Experience	Motivation	Education level	Interest	Interface	Problem
2	Low	Meduim	Meduim	High	High	Preference	Incorrect layout of widgets
3	Low	Meduim	Meduim	High	High	BOOKMARKS	Difficult navigation
4	Low	Meduim	Meduim	High	High	Regestartion	Ineffective appearance of widgets
5	Low	Meduim	Meduim	High	High	RECENT	Complicated interface
6	Meduim	Low	High	Low	Low	Preference	Incorrect layout of widgets
7	Meduim	Low	High	Low	Low	BOOKMARKS	Difficult navigation
8	Meduim	Low	High	Low	Low	Regestartion	Ineffective appearance of widgets
9	Meduim	Low	High	Low	Low	RECENT	Complicated interface
10	Meduim	Low	High	Low	Low	FindFair	Incorrect data presentation

Fig. 10 Evaluation traces example of lirbis preferences mobile interface.

It is important to note that retrieving such information is not trivial, users may or may not share some of their personal information which makes profiling them a difficult task. In addition, users are not supposed to know all defects types in the GUIs. To mitigate this, users opinions about any possible defects in the code can be extracted either explicitly through surveys like previous studies that have been conducting such human-intensive evaluation, or through semi-automated user opinion mining techniques that rely on natural language processing and deep learning to automatically extract the users opinion from comments and reviews left that can be located in the apps market. For our study, we surveyed to obtain the defects, which can be seen as the bottleneck of our approach,but this can be mitigated by using one of the opinion extraction approaches (questionnaire, interviews, ect.) [60, 61, 62]. Moreover, the main contribution of this work is to generate detection rules for a base of manually detected defects, so this manual detection is out of the scope of the papers contributions. However, all structural measures of the GUIs are done by our tool that we developed as a Java plug-in called PLAIN(PLugin for predicting the usAbility of mobile user INterface) which helps evaluator to parse the source code of GUI [87].

3.1.2 Parsing the GUI source code

This step takes as inputs the GUIs to evaluate and generates as output the evaluation metrics values. It consists of two sub-steps: 1) GUI properties extractor and 2) evaluation metrics calculator.

GUI properties extractor

This sub-step consists of parsing the GUI source code of the studied MUI to populate the evaluation metrics using the Java plug-in PLAIN. PLAIN is used to parse the source code of GUI in order to extract the graphical properties of each component of the GUI (such as the width, the height, and the alignment coordinates etc). PLAIN manages the distribution of elements in the GUI by gathering information regarding their width as the horizontal measurement taken at right angles to the left ones. The height is calculated by determining the distance from the bottom to the top of a mobile object standing upright. The alignment is the number of columns and rows used in the GUI. As an illustrative example, we are interested in extracting the properties of preferences, the previously shown GUI. Figure 11 reports its graphical properties in terms of components placements. These graphical measurements would be then used to calculate the metrics by the application of predefined formulas of each metric.

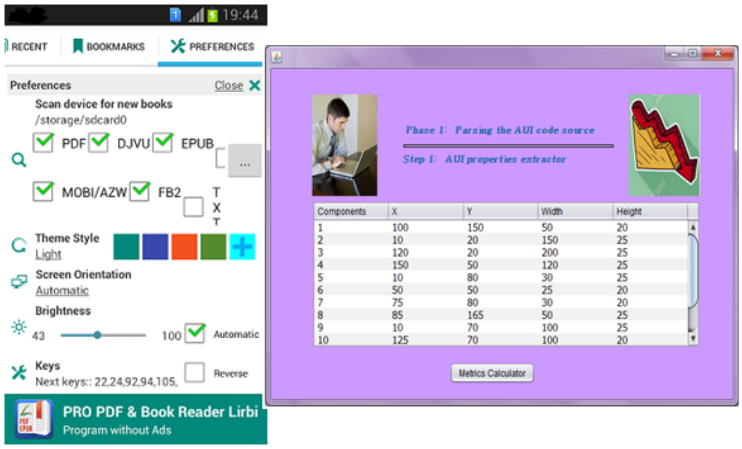

Fig. 11 properties values extracted from lirbis preferences mobile interface.

Evaluation metrics calculator

This sub-step aims to calculate the quality metrics values according to our formulas mentioned above (see section 2.4). It has as input the values of components properties and generates as output the measures of quality metrics for each GUI. The evaluation metrics are regularity, composition, sorting, complexity, integrality, density, symmetry and repartition. Figure 12 shows the values of these evaluation metrics generated by the plug-in.

These values are taken as inputs to the evolutionary algorithm in order to generate evaluation rules. In this context, we need to classify the GUI problems according to the metrics values in order to extract interesting evaluation rules. In

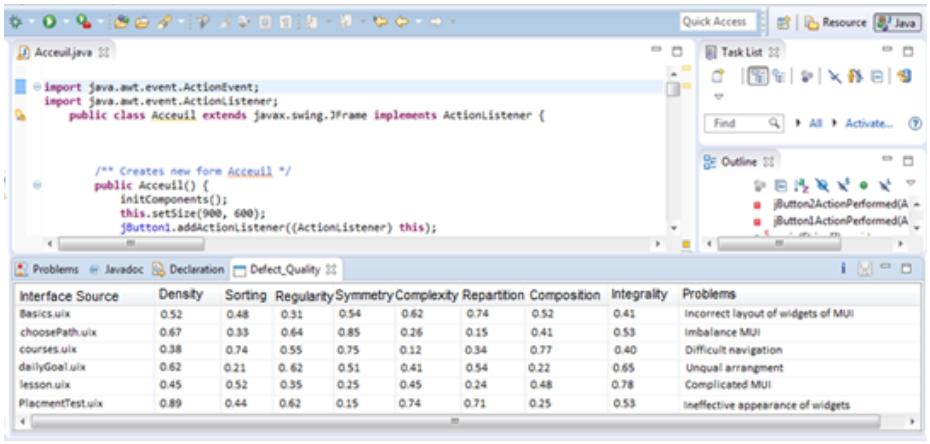

Fig. 12 Evaluation metrics values.

our study, we consider eight types of problems that correspond to each metric of GUI basing on the threshold of this metrics values. Thus, a mobile user interface with a value of the metrics lower or higher than the threshold is considered as an interface which has problems. For example, if the density metrics of an GUI has a value superior to 0.5 then the problem of this GUI is the overloaded interface. Once evaluation rules are adapted to the current GUI to evaluate, the problems can be detected.The plug-in (PLAIN) uses the adjusted evaluation rules in order to extract the quality defects of the evaluated GUIs. If a GUI has a quality problem, the detected problems can take values (high, low) according to the evaluation metrics that report this problem. However, PLAIN is a plug-in that analyzes the source code of GUIs and defects related to the quality of GUI. First, the plug-in generates all components properties for each GUI. Second, these properties are used to calculate quality evaluation metrics measures. Third, based on these measures our plug-in adjusts the evaluation rules based on the box plot technique. Finally, PLAIN analyzes the evaluation rules in order to provide a final decision about the mobile user interface to detect the problems.

3.2 Extraction of evaluation rules

The process of the extraction of evaluation rules combines randomly these inputs in order to generate interesting rules. Thus, due to a large number of context criteria, a large list of evaluation metrics and its equivalents of defect types, the creation of detection rules has become a difficult task. Therefore, it is not possible to use a determinist approach to solve this problematic [33,18,17,16,8,9,5–7]. To this end, we propose to create randomly these rules based on a heuristic search. The rule generation process aims to find the best combination between k context criteria, m quality metrics and p possible problems (defects). The number N of possible combinations is very large. With a being the number of all context criteria, b is the number of quality metrics and c is the number of possible defect types.

An exhaustive search cannot be used within a reasonable time frame to explore this huge number of combinations. The optimal solution in this huge search space is a set of rules that detect the maximum of defects (through the combination of context criteria with an associated metric) and contains the minimum of rules with the maximum number of guidance and coherence metrics. Thus, we consider the rules generation as a multi-objective optimization problem. This phase consists of generating evaluation rules. It includes four steps: 1) construction of initial population of evaluation rules (A3.1), 2) selection of best individuals (A3.2), 3) crossover of parents (A3.3) and 4) mutation of parents (A3.4). These steps performed iteratively in order to generate evaluation rules.The algorithm terminates when it reaches the last iteration accordding to the stopping criteria which is the maximum number of generation. In this phase, we need to choose a multi-objective algorithm to perform the process of evaluation rules extraction. In this context, we will compare four evolutionary algorithms: MOEA/D, NSGAII, IBEA, and SPEA2, in order to choose the appropriate algorithm that reaches the purpose of our study by satisfying the four conflicting criteria. This phase takes as inputs:

- The base of examples: contains the profiles of the users that are invited to test the GUIs and report the detected problems of GUI.
- List of context criteria: contains the various context criteria of users (age, motivation, education level, user experience, interest, etc.), the platform properties (screen size, memory, etc.) and the characteristics of the environment (luminosity, time, weather, etc.) and the possible values of each criterion.
- List of defects: contains the different type of GUI problems detected by the users.
- Metrics values: contains the values of quality metrics for each GUI calculated by the plug-in.

Besides, the quality of this solution is calculated by the fitness function that compares the different generated rules based on the base of examples. Figure 13 presents the steps of this phase.

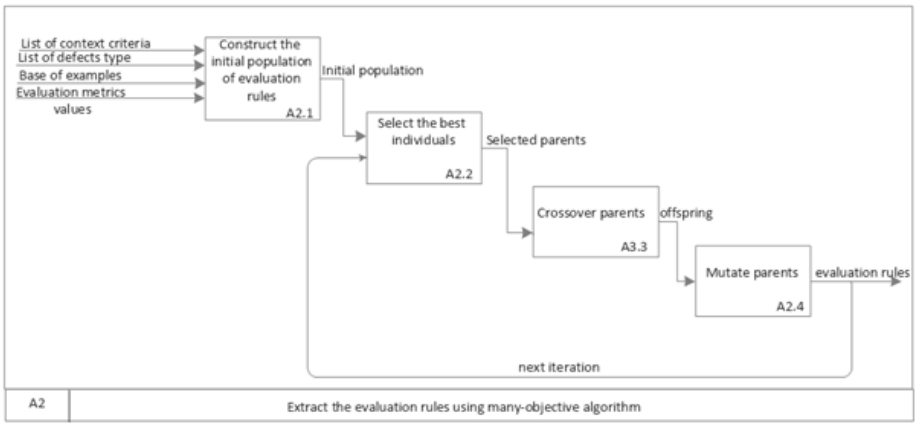

Fig. 13 Extraction of evaluation rules.

3.2.1 MOEA/D algorithm overview

MOEA/D is a multi-objective evolutionary algorithm based on decomposition. It decomposes a multi-objective optimization problem (MOP) into many single objective optimization sub-problems and optimizes them simultaneously. According to [65], MOEA/D has several features. In fact, it introduces the decompositions approaches into the multi-objective evolutionary computation. Besides, it solves multi issues of a multi-objective evolutionary algorithm such as fitness assignment and diversity maintenance because it optimizes multi subproblems rather than a whole problem. MOEA/D has multi advantages than the others evolutionary algorithm regarding complexity and solution quality. It can incorporate the objective normalization technique. The development of MOEA/D takes place around three successive stages that are selection, crossover, and mutation. In fact, it starts with the selection stage which aims to allow the best individuals of a population to reproduce. It compares them based on the values of their fitness functions. Then, the crossover stage takes place to allow the transmission of the characteristics of the best individual parent to the new individuals children by replacing the randomly chosen dimensions of the individual parent with those of another individual parent to obtain two different children. After that, there is the mutation operator that allows the modification, with a certain probability, of one or several nodes of the selected individual, to introduce some variability into the population [66]. To convert these multi-objective problems to subproblems, MOAE/D required the Tchebyche approach to reach this goal. In this approach, the objective function of the subproblem has the following form [66]:

$$\text{minimize } g^{te}(X|\lambda, K^*) = max_{1<i<m}(\lambda_i|f_i(X) - K_i^*|) \tag{9}$$

Where:
i is the subproblem.
m is the number of subproblems.
X is the solution to the subproblem i.
K is the reference point which is the vector composed of all the desired values of the objective.
λ is the weight vector.
for all i = 1,..,m $\lambda_i = 1$

During every generation, the population is a collection of the best solution found for every sub-problem. The optimal solutions of two neighboring sub-problems should be similar. Every sub-problem is optimized basically on information received from its neighboring sub-problems.

3.2.2 NSGAII overview

The non-dominated sorting genetic algorithm (NSGA-II) is a powerful heuristic search optimization method inspired by the Darwinian theory of evolution [67]. The basic idea is to explore the search space by making a population of candidate solutions, evolve toward a good solution for a specific problem. NSGA-II starts

with creating randomly the initial population P0 of individuals. Then, a child population Q0 is generated from the parent population P0 using genetic operators such as crossover and mutation. After that, a set of parents and its offsprings are assembled and a subset of individuals is selected based on the dominance principle to create the next generation. This process will be repeated until attaining the last iteration according to the stop criteria. The algorithm terminates when it reaches the stop criteria. In each iteration i, an offspring population Qt is generated from a parent population Pt using genetic operators (selection, crossover and mutation). Then, Qt and Pt are merged in order to create a global population Rt. Besides, each solution Si in the population Rt is evaluated using our four objectives (maximize the number of detected defects, minimize the solution size, maximize the guidance coverage and maximize the coherence coverage. After calculating these functions, all solutions will be sorted out in order to obtain a list of non-dominated fronts F = F1; F2; ..., with F1 is the set of non-dominated solutions, F2 the set of solutions dominated only by solutions in F1, etc. Then, the next population Pt+1 is created using the half top-ranked individuals. When the half is achieved inside a front F_i, individuals of F_i, with the same dominance, are sorted using a normalized average of the two objectives. After that genetic operators (selection, crossover, and mutation) are applied to produce the set of individuals Qt+1 to produce the population Rt+1. The algorithm terminates when it reaches the last iteration according to the stop criteria. The output of the algorithm is the set of best individuals, i.e., those in the Pareto front of the previous iteration.

3.2.3 IBEA overview

Indicator-Based evolutionary Algorithm (IBEA) is proposed by [68]. It aims to estimate the hypervolume of solution (which is the volume in the objective space covered by members of a non- dominated set of solutions). This algorithm uses binary tournaments to fill the temporary mating pool P. It implements environmental selection by iteratively removing the worst individual from the population and updating the fitness values of the remaining individuals. Initially, IBEA generates a population P of size s and sets the generation size to 0. Then, it calculates the fitness functions of individuals in P to choose an individual x from P with the smallest fitness value, and it removes x from the population and updates the fitness values of the remaining individuals. If a stopping criterion (generation size=max generations) is satisfied, then it sets x to the set of the nondominated individuals in P. Otherwise, it applies crossover and mutation operators to the mating pool P and adds the resulting offsprings to P. Then, it increments the generation size and returns to the environmental selection step.

3.2.4 SPEA2 overview

Strength Pareto Evolutionary Algorithm was proposed by [69]. SPEA uses a regular population and an archive. It starts with an initial population and an empty archive and repeatedly performs the following steps. First, it copies all non dominated population individuals to the archive; and deletes any dominated or duplicates individuals from the archive. Afterward, fitness values are assigned to both

archive and population members in which each i in the archive is attached to a strength value S (i) in [0; 1], which at the same time represents its fitness value F(i). S(i) is the number of population members j that are dominated by or equal to i concerning the objective values, divided by the population size plus one. The fitness F(j) of an individual j in the population is calculated by summing up the strength values S(i) of all archive members i that dominate or are equal to j and adding one at the end. The next step represents the mating selection phase where individuals from the union of population and archive are selected using binary tournaments. Finally, after crossover and mutation, the old population is replaced by the resulting offsprings population.

3.2.5 Adaptation of the multi-objective evolutionary algorithm

The four following subsections describe our adaptation of the evolutionary algorithm to our problem that generates a set of evaluation rules for mobile user interfaces.

Individual representation

Our work aims to extract detection rules for GUI evaluation. We can consider that our population is composed of a set of solutions (collection of rules). Our rule is presented as a set of IF-THEN rule form. Consequently, a detection rule has the following representation: IF (Context = ValueContext) AND (Metric (< / >) ValueMetric) THEN Defect type. The IF clause corresponds to the combination of a context criterion (age, motivation, etc.) with its following possible values (low, medium, high), and an evaluation metric with its threshold value using the logic operator AND. Besides, the THEN clause highlights the type of the detected defect related to the GUI quality. To present our individual, we use a vector-based solution coding. Each vector contains one rule. An example of individual representation is given in figure 14.

R1: If (age =High) AND (Density >=0.7) THEN Overloaded MUI
R2: If (Interest = Low) AND (Complexity >=0.9)THEN Complicated MUI
R3: If (Use Experience = Low) AND (Density<=0.2) THEN Overloaded MUI
R4: If (Interest = L) AND (Sorting <=0.2)THEN Icorrect_layout_of widgets

Fig. 14 Individual representation.

Generation of an initial population

The input of this stage corresponds to the list of context criteria (age, motivation, etc.), the evaluation metrics calculated by the parser (regularity, density, complexity, etc.), the base of examples and the list of possible problems types. Each attribute of the context criteria can take several values (low, medium and high) and the evaluation metrics can take a value between 0 and 1. In this stage, we extract randomly an initial population of rules from a possible combination of inputs. The used algorithm needs to encode the structure of detection rules in order to facilitate the generation of the initial population. Thus, the population is constituted of a random number of solutions, where each solution is also composed by a random set of rules.

Selection of population

At this level, we select the best solutions from the initial population in order to discover the pertinent rules. The selection phase is based on the calculated values of the fitness function. The quality of the generated solutions is evaluated using the fitness function F(x) and it is normalized in the range [0, 1]. The idea is to improve the quality of evaluation rules by reaching the four following objectives of the problem:

1. Maximizing the number of defects: we consider the best solution is the solution containing rules that maximize the number of defects.

   Q(P): is the rate of problem by solution and it is calculated as follows:

   $$Q(P) = \sum_{i=1}^{n} \frac{R(P_i)}{n} \tag{10}$$

   where R(P_i) is the Number of occurrence of problem i in solution j divided by the number of occurrence of problem i in the base of examples.
2. Minimizing the number of rules: This objective aims to minimize the rules complexity by reducing the number of rules by the solution. A high number of rules by solution does not mean that generated solution is optimal. To this end, we consider the best solution that has a minimum number of rules.
3. Maximizing the guidance coverage: we consider the best solution is the solution that maximizes the GUI guidance by satisfying the metrics related to guidance criteria (regularity, composition, sorting, complexity) by the solution.

   $$GuidanceCoverage = \sum_{i=1}^{n} (\sum_{j=1}^{nb} a_j) \tag{11}$$

   Where:

   $$\begin{cases} a(j) = 1 \text{ if the rule has a guidance metric} \\ a(j) = 0 \text{ otherwise} \end{cases} \tag{12}$$

   n=is the number of rules by solution.
   j=complexity,regularity,composition,sorting

4. Maximizing the coherence coverage: we consider the best solution is the solution that maximizes the GUI coherence by satisfying the metrics related to coherence criteria (integrality, density, symmetry, repartition) by the solution.

$$CoherenceCoverage = \sum_{i=1}^{n}(\sum_{j=1}^{nb} a_j) \tag{13}$$

Where:

$$\begin{cases} a(j) = 1 \text{ if the rule has a guidance metric} \\ a(j) = 0 \text{ otherwise} \end{cases} \tag{14}$$

n=is the number of rules by solution.
j=integrality, density, symmetry,repartition.

In our work, we consider the generation of evaluation rules as a multi-objective technique objective problem where the goal is to find the best rules maximizing the number of detected problems, minimizing rules- complexity, maximizing the coverage of GUI guidance, maximizing the coverage of GUI coherence. To this end, we will compare four evolutionary algorithms (MOAE/D, NSGAII, IBEA, and SPEA2) in order to choose the best one for automatic generation of the interesting detection rules satisfying the four conflicting criteria. Our fitness function is calculated using the following equation :

$$F(x) = \begin{cases} Minimize f_1(x) = min(nb_1, nb_2, ..., nb_n) \\ Maximize f_2(x) = Q(P) \\ Maximize f_3(x) = GuidanceCoverage \\ Maximize f_4(x) = CoherenceCoverage \end{cases} \tag{15}$$

Where x is the decision variables of the problem.

N: number of solutions by population.

nb_i: number of rules by solution i.
Guidance Coverage: the number of rules that satisfied the guidance metrics by solution.
Coherence Coverage: the number of rules that satisfied the coherence metrics by solution.

To illustrate our fitness functions, we consider 20 evaluated mobile user interfaces in which we detect three problems (workload, complex interface and regular interface). These problems occurred in the trace respectively (20 times, 15 times and 30 times). During the first generation, we have a population containing three solutions which have respectively (100 rules, 150 rules, 300 rules). The first solution has 50 rules satisfying the guidance metrics and 50 rules satisfied the coherence metrics. The second solution has 80 rules satisfying the guidance metrics and 70 rules satisfying the coherence metrics. The third solution has 200 rules satisfying the guidance metrics and 100 rules satisfying the coherence metrics. So we can

calculate our fitness function as below:

$$F(x) = \begin{cases} Minimize f_1(x) = min(100, 150, 300) \\ Maximize f_2(x) = Q(P) \\ Maximize f_3(x) = max(80, 50, 200) \\ Maximize f_4(x) = max(50, 70, 100) \end{cases} \quad (16)$$

The Q (P) of the first solution is calculated according to the following Table 2:

Table 2 Solution example

Aesthetics Defects	**Solution 1**	**Solution 2**	**Solution 3**	**base of examples**
Overloaded MUI	16	55	20	20
Complicated MUI	70	95	80	15
Difficult navigation	14	0	200	30

$$Q(P_1) = \frac{(\frac{16}{20}) + (\frac{70}{15}) + (\frac{14}{30})}{3} = 1,97 \quad (17)$$

3.3 Evaluation of MUI quality defects

This phase takes as input the evaluation rules generated by IBEA algorithm as the best algorithm compared to other algorithms (MOAE/D, NSGAII, and SPEA2) using the same number of objectives, and it generates as output the list of detected problems (defects). It is based on a java plug-in called PLAIN (Plug-ing for predicting the Adaptation quality of user INterface). It includes two steps: 1) evaluation rules adjustment (A3.1), 2) defects detection of MUI (A3.2). Figure 15 presents the structure of this phase.

3.3.1 Evaluation rules adjustment

The evaluation quality metrics are quantification mechanisms that support the examination of interface component characteristics. They are important means used to achieve the identification of quality problems of MUIs. Thus, evaluation rules combine metrics with logical operators and thresholds (see section 3.2.3.1). We assess the MUIs based on a set of evaluation rules generated by MOEA/D. In fact, the evaluation metrics measures of MUIs can be interpreted as certain symptoms of one or more defects. In the defects detection process, we need to compare these measures with an adequate threshold value. However, it is difficult to generalize these rules for all MUIs that are very different in term of a number of interfaces by application, the number of components by MUI, etc. Also, the used threshold provides no universal definition of our rules. In addition, the process of

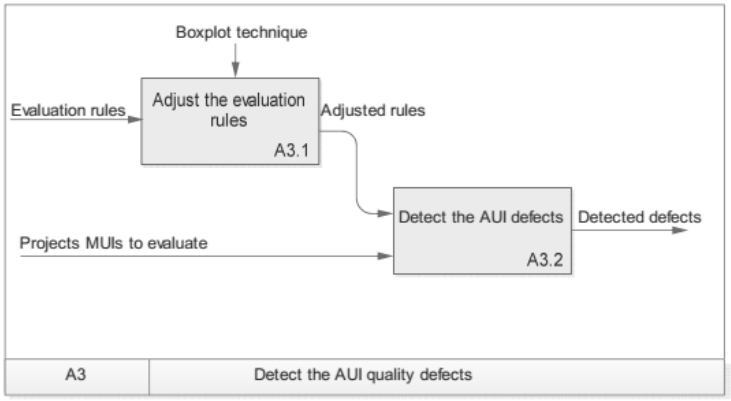

Fig. 15 Evaluation of MUI quality defects phase.

detection problems will vary depending on the selected value threshold. Increasing the value too much will cause false negatives, while decreasing it in excess will cause more false positives. So, the use of quality metrics in the problem detection process needs an adjustment mechanism to improve accuracy of proposed approach. Thus, the aim of this phase is to adjust the evaluation rules using box-plot technique. According to [45], the box-plot is a very popular graphical tool to visualize the distribution of data. Thus, it determines information about the location and the spread of the data by means of the median and the inter quartile range. In our work, box-plot takes as input the measures of quality metrics and generates as output the median of each metric that should be considered as a threshold. Figure 16 shows an example of box-plot distribution.

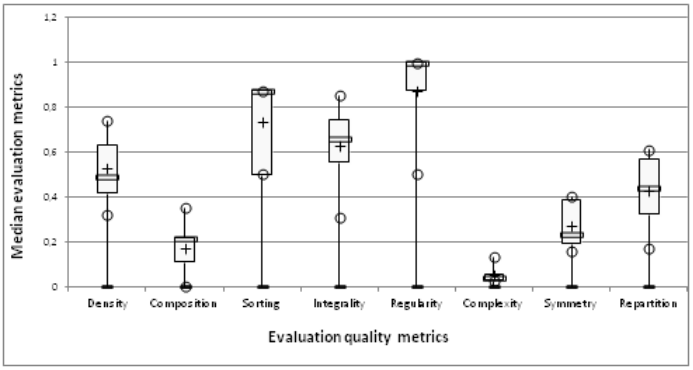

Fig. 16 The box-plot distribution of project Duolingo.

4 Validation

We conducted a set of experiments based on 24 mobile applications to measure the performance of evolutionary algorithms (MOAE/D, NSGAII, IBEA, and SPEA2) with variously given criteria. In this section, we first present our research questions and then we describe and discuss the obtained results. For the replication of our study, an implementation of the four multi-objective algorithms to detect the quality defects of mobile user interfaces can be found in [2].

4.1 Research Questions

We assess the performance of our approach by finding out whether it could generate meaningful sequences of rules that improve the quality of interfaces while reducing the number of rules needed. Our validation is conducted by addressing the following research questions outlined below. We also explain how our experiments are designed to address these questions:

Research Question 1: How do the studied evolutionary algorithms perform in comparison to the exhaustive search?

Research Question 2: To what extent can the evaluation rules cover all type of users?

Research Question 3: To what extent can the proposed approach maximize the number of detected problems of mobile user interfaces?

Research Question 4: To what extent can the proposed approach minimize the number of evaluation rules?

Research Question 5: What is the most appropriate evolutionary algorithm to use based on the performance of the four deployed algorithms (MOEA/D, NSGAII, IBEA, SPEA2)?

To answer Research Question 1, we emphasize the reason behind using a metaheuristic to solve the problem of generating the best evaluation rules for mobile user interfaces. To do so, we simulate the growth of our input problem i.e. number of mobile interfaces, and we increase the number of possible evaluation metrics to use and then we compare the performance of the exhaustive search against the four evolutionary algorithms (MOEA/D, NSGAII, IBEA, SPEA2) in terms of runtime needed to generate the rules.

To answer Research Question 2, we classify user according to the values of their context criteria into 15 category. Then we calculate the number of different rules that cover each category.

To answer Research Question 3, we evaluate the performance of the used algorithms (MOEA/D, NSGAII, IBEA, SPEA2) on the detection of eight different types of problems for the four studied projects by calculating the coverage of each studied project problem for each evaluated evolutionary algorithm.

To answer Research Question 4, we compare the results of the used algorithms (MOEA/D, NSGAII, IBEA, SPEA2), in order to extract the best algorithm which reaches the second objective (minimizing the number of used rules). We investigate the possibility of empirically achieving the same results with less number of used rules.

To answer Research Question 5, we experiment the benefit of using a multi-objective algorithm by comparing the performance of the used algorithms: MOEA/D, NSGAII, IBEA, SPEA2 in terms of assessing the developers in detecting typical defects that deteriorate the quality of interfaces.

4.2 Studied Projects

We challenge our approach by its ability to identify defects that were manually verified. To reach this purpose, the validation is being conducted over the evaluation of a benchmark of 24 open source android applications. We have chosen these projects because of their medium to large size; they considered the most popular application and can be used as input to our approach. They contain also multiple MUIs using relative techniques which make the interfaces adjusted to the screen size.

4.2.1 Experimental Setting

The evaluation of the mobile user interfaces of the studied projects should improve the quality of their applications and enhance the satisfaction of their users.

4.2.2 Subjects

The study was conducted in the Higher Institute of Management in Gabes, Tunisia. 20 students with different age, experience, and level of education, were invited to this evaluation study (55% females, 45% males). 60% are bachelor degree students and 40% are master students. The data that were collected about the participants show that only 40% of subjects have experience with software quality evaluation. However, all students have experience in using mobile applications.

4.2.3 Scenario

In our study, we use 24 mobile applications which contain 200 mobile user interfaces to evaluate. We rate it according to the user rate giving by users. The different mobile user interfaces are rated using a 7-point Likert scale. We consider that all metrics have the same importance (the weight is equal to 1).

Table 3 Properties of the studied software systems.

Mobile applications	Release	Number of interfaces
Duolingo	v3.21.0	30
Accuweather	v4.1.0	9
Easyloan calculator	v1.7.2	8
HandicraftWomen	v1.0	20
OfficeSuite	v4.0	12
transia	v1.2.1	4
Accordion	v2.4	20
Pedometre	v2.0.0	16
RangeMaker	v3.0.2	23
Resound	v1.0.5	17
VakiBank	v1.0.2	31
Video Editing	v1.0.1.40	52
VLC	v6.0.0	9
Pediatric	v2.0.1	10
Openshop	v1.1	18
AMetro	v2.0.1.4	26
AntennaPod	v1.6.1.2	24
Arzneimittel pocket	v1.4	15
Moss	v1.0	13
Math superstar	v1.0.2	11
lirbi reader	v2.3	17
Kontalk	v3.1.10	20
foodForKids	v1.0	22
ConnectBot	v1.8.6	23

4.2.4 Parameter setting

The parameter setting influences significantly the performance of search algorithms on a given search problem. It is usually difficult to preemptively set the best tuning setting. For this reason, we perform a set of experiments using several population sizes: 10, 20, 40, 80, 160, and 320 for our 4 objectives.
The maximum number of generations used is 100, 200, 400, 8000 and 1600. For each algorithm, to generate an initial population, we start by defining the maximum vector length (maximum number of rules per solution). As a higher number of operations in a solution do not necessarily mean that the results will be better, we empirically determine the best set of setting through the above-mentioned trials. Ideally, a small number of rules should be sufficient to provide a good trade-off between the fitness functions. This parameter can be also specified by the user or derived randomly from the sizes of the program and the used operations list.
During the creation, the solutions have random sizes inside the allowed range. We use the trial and error method in order to obtain a good parameter configuration. Since we are comparing different search algorithms, we classify parameters into common parameters and specific parameters.
Table 4 depicts the important common parameters. For MOEA/D, the neighborhood size is set to 20.

Table 4 The Setting of Common Parameter.

Number of objective	4	4	4	4
Algorithm Name	MOEA/D	NSGAII	IBEA	SPEA2
Population size	300	100	100	100
Number of generation	1500	1000	1000	1000
Crossover rate	0.9	0.9	0.9	0.9
Mutation rate	0.1	0.1	0.1	0.1

4.3 Results for Research Questions

4.3.1 Results for Research Question 1

During the rules generation process, our approach combines randomly evaluation structural metrics with context criteria within logical expressions (intersection AND) to create rules. In this case, the number n of possible combinations is very large. The rule generation process consists of finding the best combination of m structural metrics, k context criteria and p detected defects. In addition, for three threshold values (low, medium, high) that each metric/criterion can take, a huge number of rules can be generated. In this context, the number NR of possible combinations that have to be explored is given by:

$$PC = C_k^m * C_m^b * Cp^c = \frac{k!}{a!(k-a)!} * \frac{m!}{b!(m-b)!} * \frac{p!}{c!(p-c)!} \tag{18}$$

With a being the number of all context criteria, b is the number of quality metrics and c is the number of possible defect types.In this context, the number (NR) of generated rules will be huge and is defined by: NR = (3mkp). In such setting, the number of possible usability problems to manually illustrate with rules can be very huge. Thus, a heuristic search is needed. Table 5 shows the following experiment of considering the runtime of the brute force search, in which, all the possible combinations of rules are exhaustively explored, against the use of a meta-heuristic.

Table 5 Runtime in micro-seconds of Brute Force and used algorithms over a given number of metrics/criteria.

Number of metrics/criteria	**Brute force**	**MOEA/D**	**NSGAII**	**IBEA**	**SPEA2**
6	830	24688	25880	24561	23459
10	60000	25810	25990	24888	23698
15	14349007	27419	27541	26458	25132
20	3.49E+09	27811	27789	26888	25489
25	8.47E+11	28004	28654	27457	26457
30	2.06E+14	**28801**	**28954**	**27698**	**26895**

As noticed in Table 5, the runtime of Brute Force quickly becomes huge for a higher number of metrics. Consequently, to ensure the scalability of our solution, we considered the rule generation process as a combinatorial optimization problem. So, any solution must satisfy the predefined four objectives.
To this end, we propose an adaptation of the multi-objective evolutionary algorithm for the problem of rules generation and the results are described in the next sub-section.

4.3.2 Results for Research Question 2

In this research question, we show how the generated rules are different among different types of users. In this context, we classify users into 15 groups according to the different values of their context criteria. Table 6 presents the classification of users.

Table 6 Classification of users

Category number	**description**
C1	Users with Low age
C2	Users with medium age
C3	Users with high age
C4	Users with Low motivation
C5	Users with medium motivation
C6	Users with high motivation
C7	Users with low interest
C8	Users with medium interest
C9	Users with high interest
C10	Users with low education level
C11	Users with medium education level
C12	Users with high education level
C13	Users with low user experience
C14	Users with medium user experience
C15	Users with high user experience

After this classification we calculate the number of rules in the solution which have 'if clause' contains the value of each category. Figure 17 present the distribution of the different rules which cover all type of users.

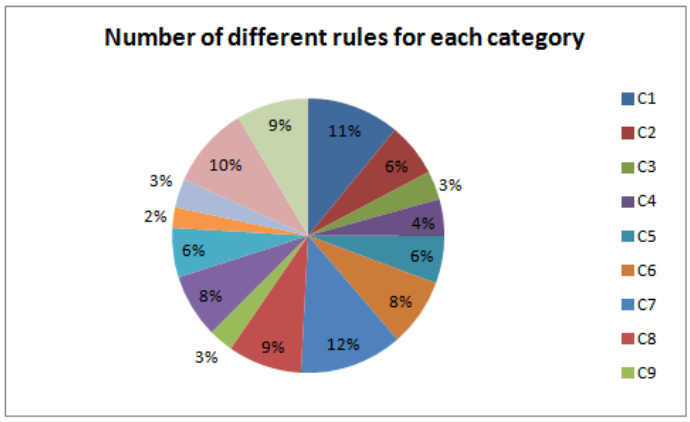

Fig. 17 The distribution of the evaluation rules in relation to the categories of users

4.3.3 Results for Research Question 3

In this section, we evaluate the performance of our used algorithms (MOEA/D, NSGAII, IBEA, SPEA2) on the detection of eight different types of MUIs defects related to guidance criteria and in the same time to coherence criteria.

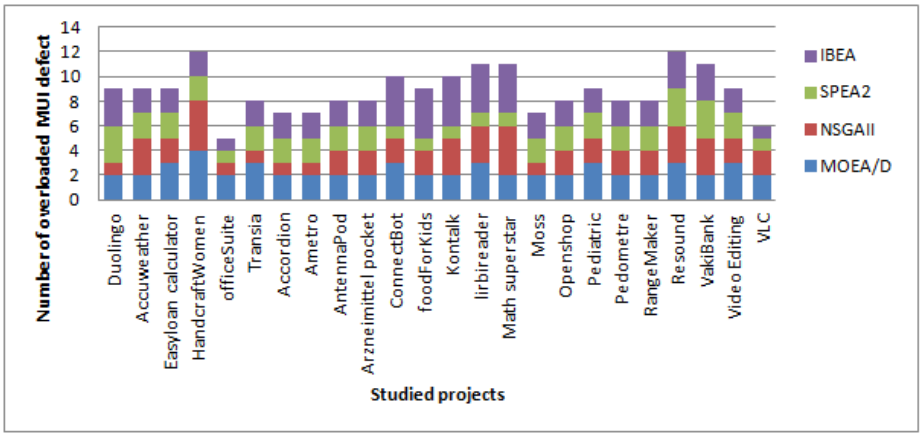

Fig. 18 The distribution of the used algorithms for the problem: Overloaded MUI

Figure 18 presents the distribution of the four used algorithms in the detection of the overloaded problem. For the studied projects, we conclude that IBEA is the best algorithm for the detection of workload problem for the five projects: libiri reader, Kontalk, foodforKids, ConnectBot and Arzneimittel pocket.

Figure 19 presents the distribution of the four used algorithms in the detection of the difficult navigation problem for mobile user interfaces. In this distribution,

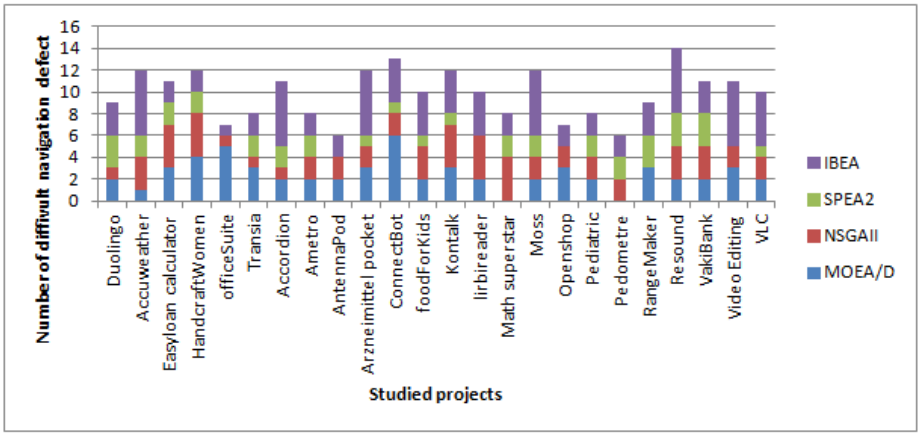

Fig. 19 The distribution of the used algorithms for the problem: Difficult navigation

we conclude that IBEA is the best algorithm for the detection of low guidance problem for the seven projects: Accuweather, Accordion, Moss, Resound, Arzneimittel pocket, Video Editing, and VLC.

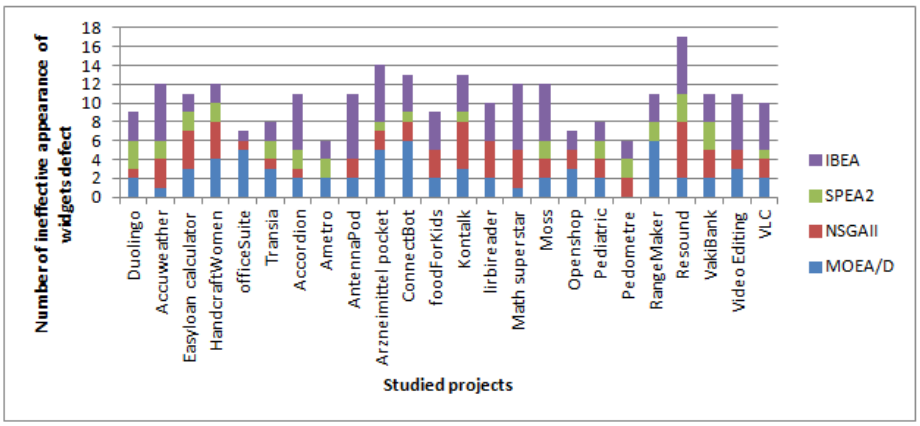

Fig. 20 The distribution of the used algorithms for the problem: Ineffective appearance of widgets

Figure 20 presents the distribution of the four used algorithms in the detection of the ineffective appearance of widgets problem for mobile user interfaces. In this distribution, we conclude that IBEA is the best algorithm for the detection of disorder interface problem for the seven projects: Accuweather, Accordion, Moss, Resound, Arzneimittel pocket, Video Editing, and VLC.

Figure 21 presents the distribution of the four used algorithms in the detection of the incorrect data presentation problem for mobile user interfaces. In this

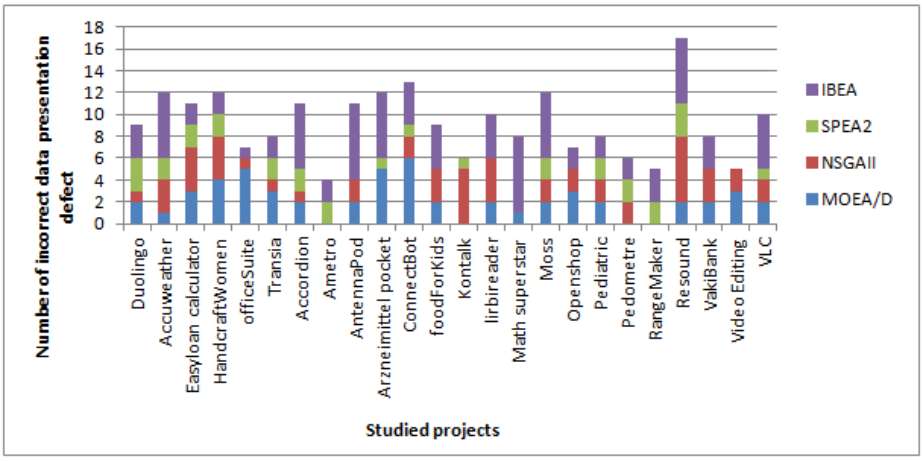

Fig. 21 The distribution of the used algorithms for the problem: Incorrect data presentation.

distribution, we conclude that this type of problem is not covered by all the used algorithm in all studied projects. However, we can note that IBEA is the best algorithm for the detection of this type of problem for the eight projects: Accuweather, Accordion, AntennaPod, Moss, Resound, Arzneimittel pocket, Math superstar and VLC.

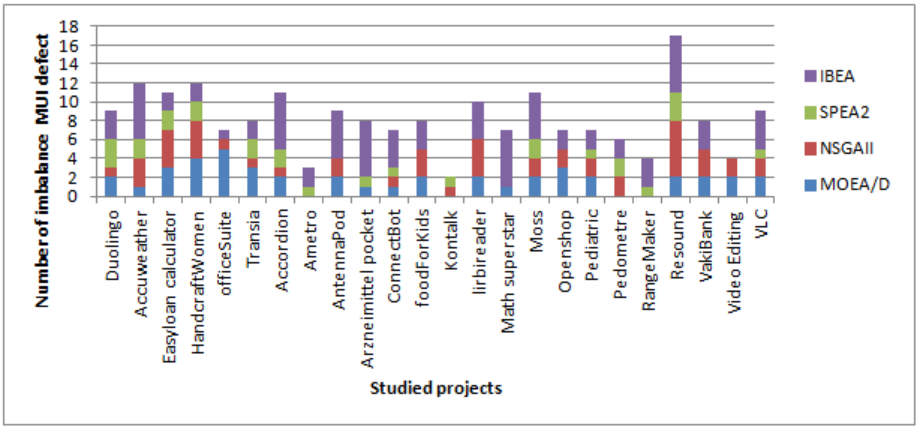

Fig. 22 The distribution of the used algorithms for the problem: Imbalance of MUI.

Figure 22 presents the distribution of the four used algorithms in the detection of the Imbalance of MUI problem for mobile user interfaces. In this distribution, we conclude that IBEA is the best algorithm for the detection of unequal arrangement problem for the ten projects.

Figure 23 presents the distribution of the four used algorithms in the detection of the incorrect layout of widgets problem for mobile user interfaces. In this distri-

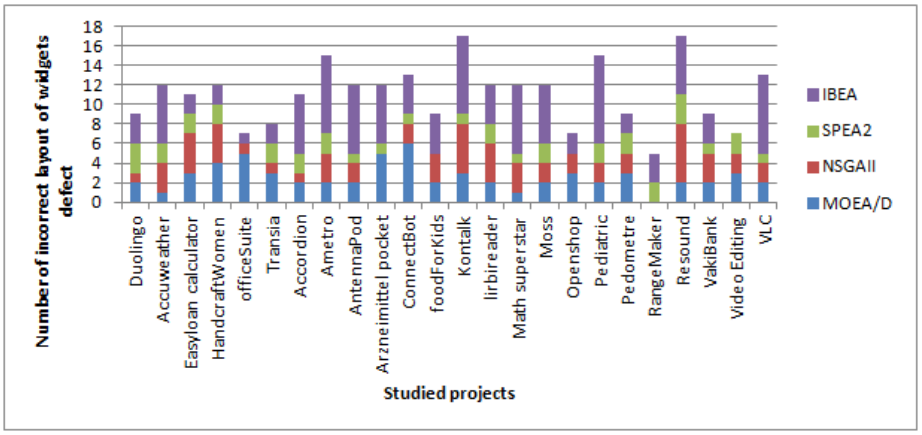

Fig. 23 The distribution of the used algorithms for the problem: Incorrect layout of widgets.

bution, we conclude that IBEA is the best algorithm for the detection of irregular interface problem for several projects at a high level.

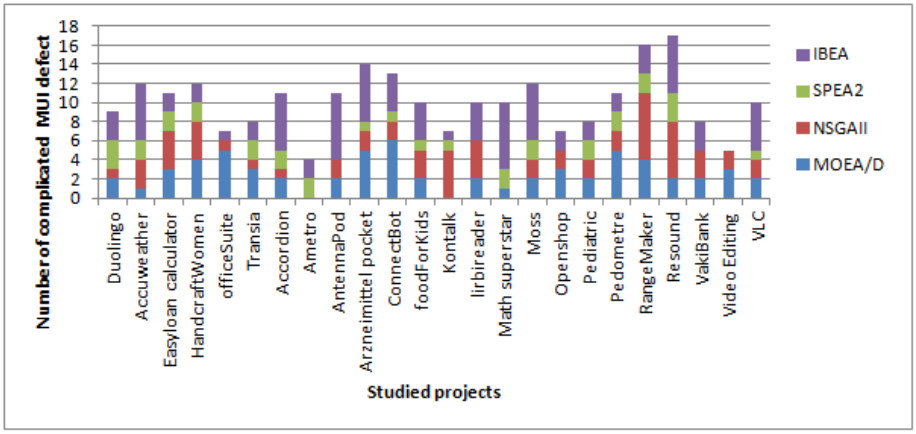

Fig. 24 The distribution of the used algorithms for the problem: Complicated MUI.

Figure 24 presents the distribution of the four used algorithms in the detection of the complicated interface problem for mobile user interfaces. In this distribution, we conclude that IBEA is the best algorithm for the detection of complex interface problem with a high level.

Figure 25 presents the distribution of the four used algorithms in the detection of the incohesion interface problem for mobile user interfaces. In this distribution, we conclude that IBEA and MOEA/D were the two best algorithms in the detection of incoherent interface problem for the most of the studied projects.
The detection rules were able to identify various types of defects in our studied projects. This ability to identify different types of defects underlines a key

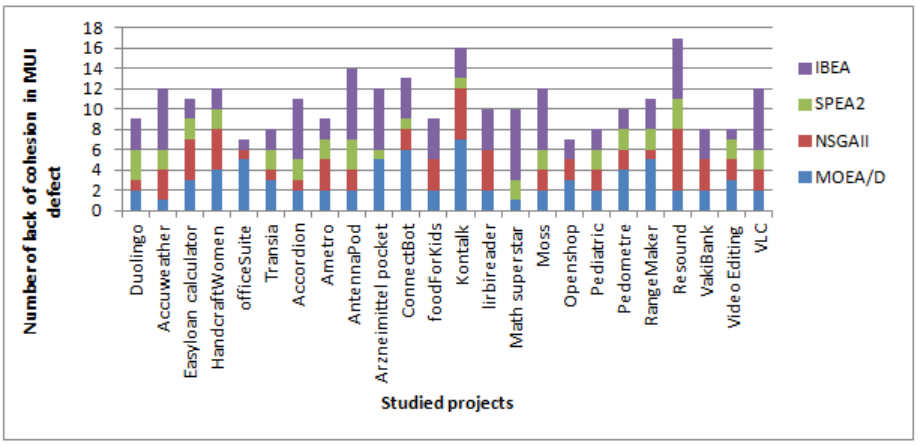

Fig. 25 The distribution of the used algorithms for the problem: Lack of cohesion in MUI.

strength of our approach. Most other existing tools and techniques rely on detecting one kind of defect, and usually, these techniques are manual, error-prone and tedious. Other automated techniques rely heavily on the structural information of the MUIs. This is reasonable considering that some defects like the Complex UI are associated with a number of modules inside the interface.

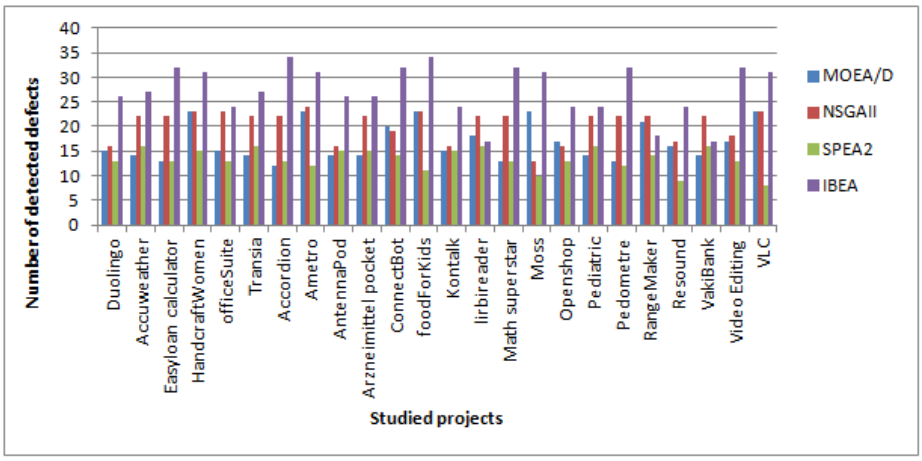

Fig. 26 Number of detected defects by project.

The mobile application Easy loan calculator was covered by all the type of defects detected on each evaluated evolutionary algorithm. However, the coverage of the other studied application out varies from one evolutionary algorithm to another with various values. Figure 26 shows that IBEA is the algorithm that detects the high number of detected problems in all the studied projects.

4.3.4 Results for Research Question 4

To test the effectiveness of the minimizing the number of rules, we consider the four used algorithms MOEA/D, NSGAII, IBEA, SPEA2). Figure 27 summarizes the results of median values of the number of generated rules over 31 independent simulation runs after applying the proposed rules by the best solution selected using the knee-point strategy.

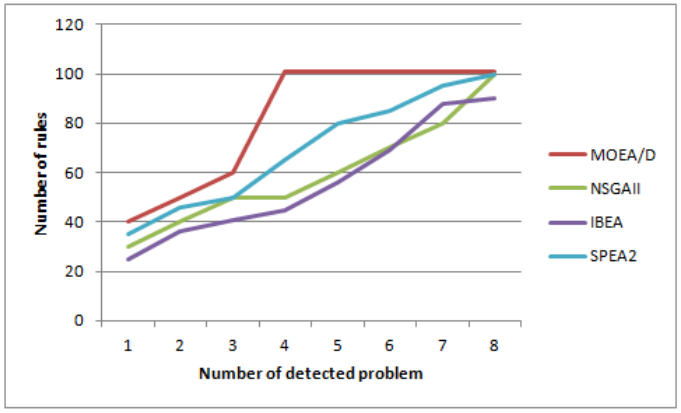

Fig. 27 The values of the number of generated rules for the used algorithms.

We notice that IBEA was able to generate less number of rules while maintaining good coverage. This can be explained by the fact that increasing the number of metrics allows the search heuristic to reach more rules combinations and so create more rules. Since the use of multi rules is not recommended as it increases the solution complexity, it is necessary for a good solution to find the trade-off between maximizing the coverage while maintaining a relatively low number of rules.

4.3.5 Results for Research Question 5

Figure 28 summarizes the results of median values of the structural metrics over 31 independent simulation runs after applying the proposed rules to the best solution selected using the knee-point strategy. The results of Figure 28 are based on the gradual consideration of all the eight metrics for the four used algorithms in terms of the second, third and fourth fitness function.

As described in Figure 28, we found that IBEA algorithm provides better structural coverage over the other algorithms in term of the three tested objectives. This is an interesting result confirming that IBEA algorithm can find very good compromises between multiple objectives that are dissimilar and it outperforms those that are produced by MOEA/D, NSGAII, and SPEA2. To assess the performance

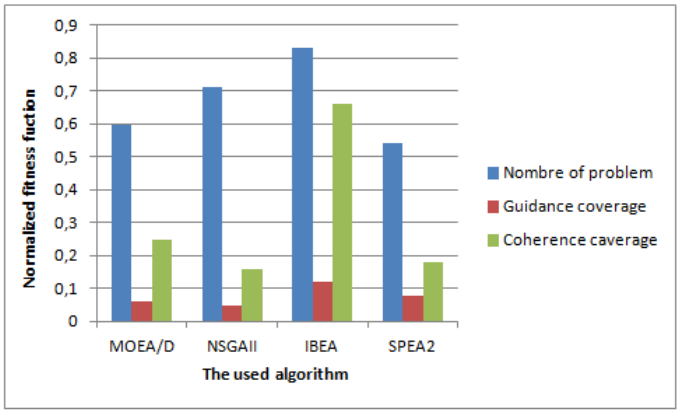

Fig. 28 Normalized fitness functions values for the used algorithm.

of all algorithms under comparison, three different issues are normally taken into account: minimize the distance of the Pareto front generated by the proposed algorithm to the exact Pareto front (GD), and to maximize the spread of solutions found (Spread), so that we can have a distribution of vectors as smooth and uniform as possible (Hypervolume).

Table 7 The performance indicators.

	MOEA/D	**NSGAII**	**IBEA**	**SPEA2**
GD	1.26+	1.73+	0.58+++	0.94+
Spread	1.56++	1.23	0.32+++	1.02+
Hypervolume	0.125++	0.023+	0.0014+++	0.032

In fact, the best algorithm is the one who has the minimum values of the performance indicators (GD, Spread, Hypervolume) along with the t-test significance. As Table 7 shows, IBEA has the minimum values of GD indicator with a value of 0.58 that is better than NSGAII with a value of 1.73, Spread Indicator with a value of 0.32 better than MOEA/D and Hypervolume indicator that is equal to 0.0014 better than SPEA2. An unpaired student t-test has been performed between each two pair of algorithms. The symbol + denotes the student t-test significance, where the number of + in each cell represents how many times the algorithm has been more significant than another algorithm with a p-value$<$0.05. As seen in Table 7, the IBEA has been more significant than all the other algorithms for the three indicators, followed by the MOEA/D, that was the second more significant in the Spread and Hypervolume indicators.

5 Threat to validity

In this section we report threats to validity to our study. As an internal validity, we have used state of the art defects that we have adapted to the context of mobile computing, these defects are known to be a the more detected problems for assessing the aesthetic quality of mobile interfaces, and we have no prior validation for these defects as we relied on their prior work for ensuring their performance. As a construct validity, we have used four evolutionary algorithms to generate detection rules, the choice of these algorithms cannot be proven to be the best choice compared to other existing algorithms, but in our approach we aim in automating the generation process and our results were statistically significant. As an external threat, we have used only 24 mobile applications and this may not be enough to generalize our findings, and thats why we are planning on extending the number of applications analyzed to challenge the scalability of our results.

6 Related work

Evaluation of mobile user interfaces aims to detect the so-called structural layout defects to improve their quality. A few methods for assessment of GUIs are proposed in the literature. In the following part, we will discuss some existing approaches related to our contribution that can be classified into (1) existing evaluation approaches, (2) existing evaluation metrics.

6.1 Existing evaluation metrics

Several metrics have been proposed to evaluate the ergonomic quality of mobile user interface [95, 84, 12]. Based on those studies, we selected several metrics for covering their complexity, their visual aesthetic, as well as their structural of the layout. (Xing et al.,) [101] defines complexity as the measure that based on the numeric size of components in the user interface, the variety of these components, and the correlation between them [101]. In this way, (Kang et al.,) [48] proposed three complexity metrics: operation complexity, transition complexity and screen complexity, to measure the complexity of user interface design. In addition, (Miyoshi et al.,) [60] evaluates the usability of screen design based on its complexity measures. The visual aesthetic is a measure of the perceived beauty of a visual stimulus [61]. The elements of visual aesthetics can be characteristics of layout, quality of graphics, amount of text, number, and choice of fonts, use of color, etc. In this context, (Ngo et al.,) [66] proposed a set of aesthetic measures for a graphical interface which are: balance, symmetry, equilibrium, sequence, order and complexity, cohesion, unity, proportion, simplicity, density, regularity, economy, homogeneity, rhythm. The values of these measures can be calculated based on the sizes and arrangements of components on the screen. Moreover, (Hartmann et al.,) [41] proposed several metrics (usability, aesthetics, memory, overall preference, engagement, service and information) of the aesthetic attribute to enhance the visual equilibrium of label layout. However, (Gonzales et al.,) [37] proposed

five aesthetic metrics: balance, linearity, orthogonality, regularity, and sequentiality. His work aims to calculate the metrics in order to assess aesthetically the graphical user interfaces. (Alemerien et al.,) [12] proposed a set of structured metrics (alignment, balance, density, size and grouping). These metrics allow evaluating the user interface based on its structure. (Sears et al.,) [82] proposed also a metric called Layout Appropriateness to organize widgets in the user interface. This metric takes as inputs a description of the sequence of widget actions in order to calculate the cost of each sequence of these actions. (Parush et al.,) [72] developed a set of metrics: size, local density, alignment, and grouping, to evaluate graphical user interfaces concerning the screen layout. However, (Constantine et al.,) [27] introduced a visual cohesion metric to assess the quality of user interface through a semantic aspect of widgets. (Shoaib et al.,) [85] proposed a coherence metric consist on three cohesion modes: low, medium and high. This metric aims to evaluate the design quality of web application.(Alemerien et al.,) [11] proposed a metric to calculate a screen layout cohesion metric (SLC metric) in order to predict the usability of user interface during the software development. Inspired from the existing evaluation metrics, we proposed a set of evaluation structural metrics devoted to assessing the quality of GUI in a different context.

6.2 Existing evaluation approaches

Due to the limited life cycle of the mobile devices and the rapid change in mobile technology, there is certainly a great demand in the mobile industry to implement evaluation techniques. These techniques aim to evaluate mobile user interfaces with time constraints and minimal efforts. However, there are several works [50, 67, 22, 40, 63] that have focused on detecting defects of mobile user interface using different methods of evaluation.

- **Final evaluation phase**
  This phase includes methods based on an experimental evaluation that can take place at the end of the development cycle of mobile applications. These methods aim to collect data about the behavior of the user in the real work situation. (Alnanihet al.,) [13] proposed a new quality-in-use model based on the international standard ISO 9126-4 (ISO/IEC TR 9126-4:2004), for measuring mobile user interface design quality. In this work, the author design theoretically valid measurement methods as a foundation for collecting and analyzing data on the new quality-in-use model of GUIs for social networking applications. In addition, they tested the model among 20 graduate students with both objective and subjective factors. (Bhandari et al.,) [23] proposed a questionnaire that combines the classical and expressive design techniques to improve mobile interface apps quality perception through the induction of arousal and valence dimension of emotion. In this work, they examined the relationship between two design factors (balance and originality) and emotional outcomes (valence and arousal). Then, they explored the impact of these affective responses on quality perceptions like pragmatic and hedonic. Hence, Coherent labs [1] developed a framework called Coherent UI Mobile. It is a

modern user interface middleware that allows developers to integrate HTML pages built with CSS and JavaScript in their game. This is achieved via the UI WebView on iOS and Androids WebView. (Su et al.,) [89] presented an intelligent user interface for reporting problematic mobile application features called QuickReview. It is developed to improve the quality and structure of mobile user interfaces reviews. The proposed QuickReview facilitates the use of mobile interface by adding reviews swiftly with ease. And, it helps developers with quick interpretation of submitted reviews by presenting a ranked list of commonly reported features.

– **Preliminary evaluation phase**
To improve the mobile user interfaces quality, one of the most used methods is the identification of defects using rules which enhance the interface presentation. (Park et al.,) [69] presents two empirical studies performed to increase the understanding of motion feedback concerning sufficient quality in mobile touchscreen user interfaces. In the first study, they examine motion properties relevant to motion feedback in mobile touchscreen user interfaces to identify their relationship with 29 affective qualities. In the second study, they explore a new factor of interactivity in mobile touchscreen user interfaces to investigate a new way to design effective qualities more effectively. These two studies are performed empirically by developing prototypes and conducting user studies to extract practical design guidelines. In addition, (Xu et al.,) [102] proposed a mobile user interface for mobile camera in order to provide the quality of the photo by enhancing the guidance of the proposed interface based on photography composition rule called rule-of-thirds. Moreover, (Kascak et al.,) [49] described and redesign the mobile user interface of remote patient monitoring (RPM) for older adults, in order to enhance the guidance of the RPM mobile application. This study uses the existing design guidelines to improve design quality of mobile health applications for older adults. In the same context, (Ruzic et al.,) [79] proposed a set of GUI design guidelines based on the Universal Design (UD) and Design for Aging strategies to ensure the guidance of mobile devices for older adults. Furthermore, coherence is also being solicited for the evaluation of mobile user interface. In fact, (Reitter et al.,) [76] demonstrated a formalism that generates coherent multimodal user interfaces, as well as its application in mobile apps. They use a generation algorithm which uses both hard constraints and scalar scores in order to cater mobile user interface adaptability.

In this way, our contribution aims to generate detection rules for MUI. In this context, our contribution aims to detect defects of MUIs by proposing an automatic tool that facilitate the detection process. Then, we use a base of example which allows generating generic rules that can be used to evaluate different types of mobile apps. In fact, instead of inviting users whenever we have an interface to evaluate, the idea is to collect for once a base of examples that would be used to generate generic detection rules. These rules can be exploited for the evaluation of different mobile apps.

7 Conclusion and future work

The mobile user interfaces have been in ever-increasing development. In this context, several research initiatives have been proposed to model the context of use, to design and generate the mobile user interface [35,68]. However, there are few studies about the evaluation of mobile user interfaces. One of the widely used methods to detect defects is applying standard detection rules which have multi-effects such as time-consuming, error-prone task. In this context, we proposed an approach that incorporates the user's feedback and profile when assessing the quality of the mobile user interface.

To this end, we consider the generation of evaluation rules as a Multi-objective optimization problem where the goal is to find the best rules maximizing the number of detected defects, minimizing rules-complexity, maximizing the coverage MUI guidance and maximizing the coverage of MUI coherence. Therefore, we tested various Multi-Objective Evolutionary Algorithms to automatically generate the best detection rules satisfying the four conflicting criterias. Our approach takes as input a base of examples, a set of context criteria (experience, study level, motivation, etc.), a list of possible problem types (workload, low guidance, complexity, etc.), a set of quality metrics (density, grouping, regularity etc.) and generates as output an exciting detection rules. After that, we will use these rules to detect quality defects of MUI.

However, there are three significant limitations to our work. Thus, the performance of our approach depends on the availability of mobile user interfaces examples. In our work, we use a benchmarked of mobile android applications to extract evaluation rules. So, we plan to expand our base of example by additional interfaces to detect more different problems of mobile user interfaces. For the generated rules, the representation of such rule depends on one context criterion such as (age, motivation, education level, user experience, and interest) and one of the proposed metrics that correlate semantically with this criterion. On the other hand, we can combine some metrics that relate with the same context criterion to improve the detection of evaluation rules.

Another limitation of the proposed approach is the number of objectives. Indeed, we use a multi-objective evolutionary algorithm that performs to optimize a multi-objective problem. In fact, we can increase the number of objectives to improve the performance of our approach. One of our perspectives is the extraction of refactoring rules for the problem detection to mend the quality of the studied mobile interfaces.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

References

1. Coherent labs. URL http://coherent-labs.com/product-coherent-ui/
2. expriment. URL https://github.com/mabroukachouchane/master-Jmetal
3. Iso. URL https://www.iso.org/obp/ui/iso:std:iso:9241:-11:ed-2:v1:en
4. Abrahão, S., Iborra, E., Vanderdonckt, J.: Usability evaluation of user interfaces generated with a model-driven architecture tool. In: Maturing Usability, pp. 3–32. Springer (2008)
5. Abualigah, L.M., Khader, A.T.: Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. The Journal of Supercomputing **73**(11), 4773–4795 (2017)
6. Abualigah, L.M., Khader, A.T., Hanandeh, E.S.: Hybrid clustering analysis using improved krill herd algorithm. Applied Intelligence **48**(11), 4047–4071 (2018)
7. Abualigah, L.M., Khader, A.T., Hanandeh, E.S.: A new feature selection method to improve the document clustering using particle swarm optimization algorithm. Journal of Computational Science **25**, 456–466 (2018)
8. Abualigah, L.M.Q.: Feature Selection and Enhanced Krill Herd Algorithm for Text Document Clustering. Springer (2019)
9. Abualigah, L.M.Q., Hanandeh, E.S.: Applying genetic algorithms to information retrieval using vector space model. International Journal of Computer Science, Engineering and Applications **5**(1), 19 (2015)
10. Akiki, P.A., Bandara, A.K., Yu, Y.: Adaptive model-driven user interface development systems. ACM Computing Surveys (CSUR) **47**(1), 9 (2014)
11. Alemerien, K., Magel, K.: Slc: a visual cohesion metric to predict the usability of graphical user interfaces. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 1526–1533. ACM (2015)
12. Alemerien, K.A.: Metrics and tools to guide design of graphical user interfaces. Ph.D. thesis, North Dakota State University (2014)
13. Alnanih, R., Ormandjieva, O., Radhakrishnan, T.: A new quality-in-use model for mobile user interfaces. In: Software Measurement and the 2013 Eighth International Conference on Software Process and Product Measurement (IWSM-MENSURA), 2013 Joint Conference of the 23rd International Workshop on, pp. 165–170. IEEE (2013)
14. Aquino, N., Vanderdonckt, J., Condori-Fernández, N., Dieste, Ó., Pastor, Ó.: Usability evaluation of multi-device/platform user interfaces generated by model-driven engineering. In: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, p. 30. ACM (2010)
15. Arabshian, K., Schulzrinne, H.: Distributed context-aware agent architecture for global service discovery. In: The Second International Workshop on Semantic Web Technology For Ubiquitous and Mobile Applications (SWUMA06) (2006)
16. Arqub, O.A., Abo-Hammour, Z.: Numerical solution of systems of second-order boundary value problems using continuous genetic algorithm. Information sciences **279**, 396–415 (2014)
17. Arqub, O.A., Al-Smadi, M., Momani, S., Hayat, T.: Application of reproducing kernel algorithm for solving second-order, two-point fuzzy boundary value problems. Soft Computing **21**(23), 7191–7206 (2017)
18. Arqub, O.A., Mohammed, A.S., Momani, S., Hayat, T.: Numerical solutions of fuzzy differential equations using reproducing kernel hilbert space method. Soft Computing **20**(8), 3283–3302 (2016)
19. Baldwin, T., Chai, J.: Towards online adaptation and personalization of key-target resizing for mobile devices. In: Proceedings of the 2012 ACM international conference on Intelligent User Interfaces, pp. 11–20. ACM (2012)

20. Bao, T., Cao, H., Chen, E., Tian, J., Xiong, H.: An unsupervised approach to modeling personalized contexts of mobile users. Knowledge and Information Systems **31**(2), 345–370 (2012)
21. Bastien, J.C., Scapin, D.L.: Ergonomic criteria for the evaluation of human-computer interfaces. Ph.D. thesis, Inria (1993)
22. Bertini, E., Gabrielli, S., Kimani, S.: Appropriating and assessing heuristics for mobile computing. In: Proceedings of the working conference on Advanced visual interfaces, pp. 119–126. ACM (2006)
23. Bhandari, U., Neben, T., Chang, K., Chua, W.Y.: Effects of interface design factors on affective responses and quality evaluations in mobile applications. Computers in Human Behavior **72**, 525–534 (2017)
24. Biel, B., Grill, T., Gruhn, V.: Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application. Journal of Systems and Software **83**(11), 2031–2044 (2010)
25. Brusilovsky, P., Chavan, G., Farzan, R.: Social adaptive navigation support for open corpus electronic textbooks. In: International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, pp. 24–33. Springer (2004)
26. Cámara, J., de Lemos, R., Laranjeiro, N., Ventura, R., Vieira, M.: Testing the robustness of controllers for self-adaptive systems. Journal of the Brazilian Computer Society **20**(1), 1 (2014)
27. Constantine, L.L.: Visual coherence and usability: a cohesion metric for assessing the quality of dialogue and screen designs. In: Computer-Human Interaction, 1996. Proceedings., Sixth Australian Conference on, pp. 115–121. IEEE (1996)
28. DeJong, M., Schellens, P.J.: Reader-focused text evaluation: An overview of goals and methods. Journal of business and technical communication **11**(4), 402–432 (1997)
29. Dey, A.K., Abowd, G.D.: Cybreminder: A context-aware system for supporting reminders. In: International Symposium on Handheld and Ubiquitous Computing, pp. 172–186. Springer (2000)
30. Ehmke, C., Wilson, S.: Identifying web usability problems from eye-tracking data. In: Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI... but not as we know it-Volume 1, pp. 119–128. British Computer Society (2007)
31. Erfani, M., Zandi, M., Rilling, J., Keivanloo, I.: Context-awareness in the software domaina semantic web enabled modeling approach. Journal of Systems and Software **121**, 345–357 (2016)
32. Følstad, A., Hornbæk, K.: Work-domain knowledge in usability evaluation: Experiences with cooperative usability testing. Journal of systems and software **83**(11), 2019–2030 (2010)
33. Geem, Z.W., Kim, J.H.: Application of computational intelligence techniques to an environmental flow formula. International Journal of Fuzzy Logic and Intelligent Systems **18**(4), 237–244 (2018)
34. Gena, C.: Methods and techniques for the evaluation of user-adaptive systems. The knowledge engineering review **20**(1), 1–37 (2005)
35. Gena, C., Weibelzahl, S.: Usability engineering for the adaptive web, the adaptive web: methods and strategies of web personalization (2007)
36. Ghiani, G., Polet, J., Antila, V., Mäntyjärvi, J.: Evaluating context-aware user interface migration in multi-device environments. Journal of Ambient Intelligence and Humanized Computing **6**(2), 259–277 (2015)
37. González, S., Montero, F., González, P.: Balores: a suite of principles and metrics for graphical user interface evaluation. In: Proceedings of the 13th International Conference on Interacción Persona-Ordenador, p. 9. ACM (2012)
38. Hariri, B., Shirmohammadi, S., Pakravan, M.R.: A distributed interest management scheme for massively multi-user virtual environments. In: Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2008. VECIMS 2008. IEEE Conference on, pp. 111–115. IEEE (2008)
39. Hariri, B., Shirmohammadi, S., Pakravan, M.R., Alavi, M.H.: An adaptive latency mitigation scheme for massively multiuser virtual environments. Journal of Network and Computer Applications **32**(5), 1049–1063 (2009)
40. Harrison, R., Flood, D., Duce, D.: Usability of mobile applications: literature review and rationale for a new usability model. Journal of Interaction Science **1**(1), 1 (2013)

41. Hartmann, K., Götzelmann, T., Ali, K., Strothotte, T.: Metrics for functional and aesthetic label layouts. In: International Symposium on Smart Graphics, pp. 115–126. Springer (2005)
42. Hellmann, T.D., Maurer, F.: Rule-based exploratory testing of graphical user interfaces. In: 2011 Agile Conference, pp. 107–116. IEEE (2011)
43. Hobbs, J., Pan, F.: Time ontology in owl. ontology engineering patterns task force of the semantic web best practices and deployment working group. World Wide Web Consortium (W3C) notes (2006)
44. Huart, J., Kolski, C., Bastien, C.: L'évaluation de documents multimédias, état de l'art. Objectiver l'humain **1**, 211–250 (2008)
45. Hubert, M., Vandervieren, E.: An adjusted boxplot for skewed distributions. Computational statistics & data analysis **52**(12), 5186–5201 (2008)
46. Hwang, W., Salvendy, G.: Number of people required for usability evaluation: the 10±2 rule. Communications of the ACM **53**(5), 130–133 (2010)
47. Ines, G., Makram, S., Mabrouka, C., Mourad, A.: Evaluation of mobile interfaces as an optimization problem. Procedia Computer Science **112**, 235–248 (2017)
48. Kang, H.G., Seong, P.H.: An information theory-based approach for quantitative evaluation of user interface complexity. IEEE Transactions on Nuclear Science **45**(6), 3165–3174 (1998)
49. Kascak, L.R., Rébola, C.B., Sanford, J.A.: Integrating universal design (ud) principles and mobile design guidelines to improve design of mobile health applications for older adults. In: Healthcare Informatics (ICHI), 2014 IEEE International Conference on, pp. 343–348. IEEE (2014)
50. Kjeldskov, J., Stage, J.: New techniques for usability evaluation of mobile systems. International journal of human-computer studies **60**(5-6), 599–620 (2004)
51. Kobsa, A.: Privacy-enhanced personalization. Communications of the ACM **50**(8), 24–33 (2007)
52. Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., Malm, E.J.: Managing context information in mobile devices. IEEE pervasive computing **2**(3), 42–51 (2003)
53. Lee, H., Choi, Y.S., Kim, Y.J.: An adaptive user interface based on spatiotemporal structure learning. IEEE Communications Magazine **49**(6) (2011)
54. Lelli, V., Blouin, A., Baudry, B.: Classifying and qualifying gui defects. In: Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on, pp. 1–10. IEEE (2015)
55. Lin, X., Li, S., Xu, J., Shi, W., Gao, Q.: An efficient context modeling and reasoning system in pervasive environment: Using absolute and relative context filtering technology. In: International Conference on Web-Age Information Management, pp. 357–367. Springer (2005)
56. Liu, B.F., Chou, S.C., Lin, Y.T., Lin, J.Y.: Toward easy delivery of device-oriented adaptive user interface on mobile devices. In: Information Science and Service Science (NISS), 2011 5th International Conference on New Trends in, vol. 1, pp. 80–85. IEEE (2011)
57. Magoulas, G.D., Chen, S.Y., Papanikolaou, K.A.: Integrating layered and heuristic evaluation for adaptive learning environments. In: Proceedings of the Second Workshop on Empirical Evaluation of Adaptive Systems, held at the 9th International Conference on User Modeling UM2003, Pittsburgh, pp. 5–14 (2003)
58. Marinescu, R.: Detection strategies: Metrics-based rules for detecting design flaws. In: Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on, pp. 350–359. IEEE (2004)
59. Masra, S.M.W., Goh, K., Muhammad, M.S., Djojodibroto, R.D., Sapawi, R., Kipli, K., Shahrom, N.S.: Graphical user interface (gui) of digital index evaluation system for finger clubbing identification. Scientific Research Journal **14**(2), 99–112 (2017)
60. Miyoshi, T., Murata, A.: Input device using eye tracker in human-computer interaction. In: Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on, pp. 580–585. IEEE (2001)
61. Moorthy, A.K., Bovik, A.C.: Blind image quality assessment: From natural scene statistics to perceptual quality. IEEE transactions on Image Processing **20**(12), 3350–3364 (2011)
62. Moran, K., Bernal-Cárdenas, C., Curcio, M., Bonett, R., Poshyvanyk, D.: Machine learning-based prototyping of graphical user interfaces for mobile apps. arXiv preprint arXiv:1802.02312 (2018)

63. Moumane, K., Idri, A., Abran, A.: Usability evaluation of mobile applications using iso 9241 and iso 25062 standards. SpringerPlus **5**(1), 548 (2016)
64. Myers, A.C.: Bidirectional object layout for separate compilation. ACM SIGPLAN Notices **30**(10), 124–139 (1995)
65. Neil, T.: Mobile design pattern gallery: UI patterns for smartphone apps. " O'Reilly Media, Inc." (2014)
66. Ngo, D., Teo, L., Byrne, J.: Formalising guidelines for the design of screen layouts. Displays **21**(1), 3–15 (2000)
67. O'Malley, C., Vavoula, G., Glew, J., Taylor, J., Sharples, M., Lefrere, P., Lonsdale, P., Naismith, L., Waycott, J.: Guidelines for learning/teaching/tutoring in a mobile environment (2005)
68. Paramythis, A., Weibelzahl, S., Masthoff, J.: Layered evaluation of interactive adaptive systems: framework and formative methods. User Modeling and User-Adapted Interaction **20**(5), 383–453 (2010)
69. Park, D., Lee, J.H., Kim, S.: Investigating the affective quality of interactivity by motion feedback in mobile touchscreen user interfaces. International Journal of Human-Computer Studies **69**(12), 839–853 (2011)
70. Park, J., Han, S.H., Kim, H.K., Cho, Y., Park, W.: Developing elements of user experience for mobile phones and services: survey, interview, and observation approaches. Human Factors and Ergonomics in Manufacturing & Service Industries **23**(4), 279–293 (2013)
71. Parra-Arnau, J., Rebollo-Monedero, D., Forné, J.: Measuring the privacy of user profiles in personalized information systems. Future Generation Computer Systems **33**, 53–63 (2014)
72. Parush, A., Nadir, R., Shtub, A.: Evaluating the layout of graphical user interface screens: Validation of a numerical computerized model. International Journal of Human-Computer Interaction **10**(4), 343–360 (1998)
73. Paterno, F.: Model-based design and evaluation of interactive applications. Springer Science & Business Media (2012)
74. Pombinho, P., Carmo, M.B., Afonso, A.P.: Adaptive mobile visualization-the chameleon framework. Computer Science and Information Systems **12**(2), 445–464 (2015)
75. Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., De Bosschere, K.: Towards an extensible context ontology for ambient intelligence. In: European Symposium on Ambient Intelligence, pp. 148–159. Springer (2004)
76. Reitter, D., Panttaja, E.M., Cummins, F.: Ui on the fly: Generating a multimodal user interface. In: Proceedings of HLT-NAACL 2004: Short Papers, pp. 45–48. Association for Computational Linguistics (2004)
77. Riegler, A., Holzmann, C.: Measuring visual user interface complexity of mobile applications with metrics. Interacting with Computers **30**(3), 207–223 (2018)
78. Rousseau, B., Browne, P., Malone, P., Ó Foghlú, M.: User profiling for content personalisation in information retrieval (2004)
79. Ruzic, L., Lee, S.T., Liu, Y.E., Sanford, J.A.: Development of universal design mobile interface guidelines (udmig) for aging population. In: International Conference on Universal Access in Human-Computer Interaction, pp. 98–108. Springer (2016)
80. Schmidt, A., Beigl, M., Gellersen, H.W.: There is more to context than location. Computers & Graphics **23**(6), 893–901 (1999)
81. Schmidt, B., Galar, D., Wang, L.: Context awareness in predictive maintenance. In: Current trends in reliability, availability, maintainability and safety, pp. 197–211. Springer (2016)
82. Sears, A.: Layout appropriateness: A metric for evaluating user interface widget layout. IEEE Transactions on Software Engineering **19**(7), 707–719 (1993)
83. Shitkova, M., Holler, J., Heide, T., Clever, N., Becker, J.: Towards usability guidelines for mobile websites and applications. In: Wirtschaftsinformatik, pp. 1603–1617 (2015)
84. Shneiderman, B., Plaisant, C.: The future of graphic user interfaces: Personal role managers. In: BCS HCI, pp. 3–8 (1994)
85. Shoaib, M., Shah, A., Majeed, F.: Software design quality metrics for web based applications. Pakistan Journal of science **63**(1) (2011)

86. Soui, M., Abed, M., Kolski, C., Ghèdira, K.: Evaluation by simulation to optimise information systems personalisation quality in logistics. International Journal of Production Research **50**(13), 3579–3593 (2012)
87. Soui, M., Chouchane, M., Gasmi, I., Mkaouer, M.W.: Plain: Plugin for predicting the usability of mobile user interface. In: VISIGRAPP (1: GRAPP), pp. 127–136 (2017)
88. Stephanidis, C., Paramythis, A., Sfyrakis, M.: Evaluating adaptable and adaptive user interfaces: lessons learned from the development of the avanti web browser. In: 5th ERCIM Workshop on" User Interfaces for All"(pp. 22.1-22.6) (1999)
89. Su'a, T., Licorish, S.A., Savarimuthu, B.T.R., Langlotz, T.: Quickreview: A novel data-driven mobile user interface for reporting problematic app features. In: Proceedings of the 22nd International Conference on Intelligent User Interfaces, pp. 517–522. ACM (2017)
90. Taconet, C., Kazi-Aoul, Z.: Context-awareness and model driven engineering: Illustration by an e-commerce application scenario. In: Digital Information Management, 2008. ICDIM 2008. Third International Conference on, pp. 864–869. IEEE (2008)
91. Terdjimi, M., Médini, L., Mrissa, M.: Towards a meta-model for context in the web of things. In: Karlsruhe Service Summit Workshop (2016)
92. Thalmann, S.: Adaptation criteria for the personalised delivery of learning materials: A multi-stage empirical investigation. Australasian Journal of Educational Technology **30**(1) (2014)
93. Thevenin, D., Coutaz, J.: Plasticity of user interfaces: Framework and research agenda. In: Interact, vol. 99, pp. 110–117 (1999)
94. Van Velsen, L., Van Der Geest, T., Klaassen, R., Steehouder, M.: User-centered evaluation of adaptive and adaptable systems: a literature review. The knowledge engineering review **23**(3), 261–281 (2008)
95. Vanderdonckt, J., Gillo, X.: Visual techniques for traditional and multimedia layouts. In: Proceedings of the workshop on Advanced visual interfaces, pp. 95–104. ACM (1994)
96. Vos, T.E., Kruse, P.M., Condori-Fernández, N., Bauersfeld, S., Wegener, J.: Testar: Tool support for test automation at the user interface level. International Journal of Information System Modeling and Design (IJISMD) **6**(3), 46–83 (2015)
97. Walker, T.: State of the us internet in q1 2012. ComScore Inc. State of the Internet: US Quarter One (2012)
98. Wang, X.H., Zhang, D.Q., Gu, T., Pung, H.K.: Ontology based context modeling and reasoning using owl. In: Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on, pp. 18–22. Ieee (2004)
99. von Wangenheim, C.G., Porto, J.V.A., Hauck, J.C., Borgatto, A.F.: Do we agree on user interface aesthetics of android apps? arXiv preprint arXiv:1812.09049 (2018)
100. Weiβenberg, N., Voisard, A., Gartmann, R.: Using ontologies in personalized mobile applications. In: Proceedings of the 12th annual ACM international workshop on Geographic information systems, pp. 2–11. ACM (2004)
101. Xing, J., Manning, C.A.: Complexity and automation displays of air traffic control: Literature review and analysis. Tech. rep., FEDERAL AVIATION ADMINISTRATION OKLAHOMA CITY OK CIVIL AEROMEDICAL INST (2005)
102. Xu, Y., Ratcliff, J., Scovell, J., Speiginer, G., Azuma, R.: Real-time guidance camera interface to enhance photo aesthetic quality. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, pp. 1183–1186. ACM (2015)
103. Ye, X., Bunescu, R., Liu, C.: Mapping bug reports to relevant files: A ranking model, a fine-grained benchmark, and feature evaluation. IEEE Transactions on Software Engineering **42**(4), 379–402 (2016)
104. Zen, M., Vanderdonckt, J.: Towards an evaluation of graphical user interfaces aesthetics based on metrics. In: Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on, pp. 1–12. IEEE (2014)