Evaluation of Mobile Interfaces as an Optimization Problem

Ines Gasmi, Makram Soui, Mabrouka Chouchane, Mourad Abed

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 112 (2017) 235–248

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Knowledge Based and Intelligent Information and Engineering Systems, KES2017, 6-8 September 2017, Marseille, France

Evaluation of Mobile Interfaces as an Optimization Problem

Gasmi Ines[a*], Soui Makram[a], Chouchane Mabrouka[a], Abed Mourad[b]

[a]*University of Manouba, Tunis, Tunisia*
[b] *UVHC, LAMIH-UMR CNRS 8201, F-59313 Valenciennes, France*

Abstract

Mobile applications are more and more present everywhere (at home, at work, in public places, etc.). Many academic and industrial studies are conducted about design methods and tools for mobile user interface generation. However, the evaluation of such interfaces is object of relatively few propositions and studies in the literature. The existing evaluation methods are widely based on a questionnaire, survey, eye tracking, etc. to assess mobile interface. These methods are time-consuming, error-prone task. In fact, one of the widely used methods to assess quality of MUI is using detection rules. But, the manual definition of these methods is still a difficult task. In this context, we define a method that generates evaluation rules for assessing the quality of mobile interfaces. To this end, we consider the generation of evaluation rules as a mono-objective technique problem where the goal is to find the best rules maximizing the quality of mobile interfaces. We evaluate our approach on four mobile applications. This study was designed around the android mobile devices. The obtained results confirm the efficiency of our technique with an average of more than 70% of precision and recall.

Peer-review under responsibility of KES International

Keywords: Mobile user interface, Evaluation, Defects detection, Genetic programming;

1. Introduction

Nowadays, the diversity of context in which the interaction between system and users takes place is a new challenge for developers. In fact, the mobile and tablet devices sales are growing every day and there is an increased attention to mobile user interfaces. In this way, there are widely innovations and development tools of new mobile services. In this context, mobile interfaces are designed to be used on smartphones and tablet computers. These mobile interfaces can be used in everyday life and/or at work, in different contexts such as e-learning, transport, games, social networking, weather, etc.

*Corresponding E-mail address: gasmioines@gmail.com
souii_makram@yahoo.fr
chouchane.mabrouka@gmail.com
Mourad.Abed@univ-valenciennes.fr

1877-0509 © 2017 The Authors. Published by Elsevier B.V.
Peer-review under responsibility of KES International
10.1016/j.procs.2017.08.234

According to Statistics, the number of mobile applications downloads worldwide in 2009 is 2.52 billion. In addition, the social network and e-commerce development stimulates the evolution of the mobile operating systems (Android, Windows phone, iOS, etc.). For example, the number of application available in Google play store is 1.600.000 and the number of applications in windows phone store is 340.000. As mobile technologies became widespread, users will be more motivated to use more portable devices and interacting with mobile applications. Indeed, 3600 million of users are subscribed in mobile services since 2014[21].

A mobile user interface (MUI) is the graphical and usually touch sensitive display on a mobile device, such as a smartphone or tablet that allows the user to interact with the device's apps, features, content and functions. User interface level represent 50% of software code which proves the importance of this level in the correctness and the effectiveness of the application[16, 18]. Hellmann[8] reported that MUI-related defects have a significant impact on the end users of system. 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 mobile UI is very important phase in the development to decrease the maintenance cost of application.

Several studies have been carried out to model the user as well as to design methods and tools for mobile UI generation. However, few evaluation studies have been proposed for mobile user interface quality assessment. The literature has shown that there are many evaluation techniques to assess GUI such as heuristic evaluation, questionnaire, interview, etc. However, there is no consensus as how mobile interface should be assessed. In fact, mobile UI has limited characteristics (small screen size, event centric, simple and Intuitive GUI) compared to traditional desktop GUI which need specific evaluation techniques. The diversity of mobile platforms makes the evaluation task very difficult to ensure that mobile interfaces can be used effectively and efficiently under any context and environment. Thus, it is necessary to envisage new methods that take into account the evaluation of mobile interface. In this context, we propose in this paper an evaluation approach based on the genetic programming algorithm. Our approach aims to find the best evaluation rules that maximize the quality of mobile user interface. These rules can be used to detect defects of mobile user interface.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 gives an overview of evaluation of MUI. Section 4 gives an overview of our proposal and explains how we adapted the GP algorithm to find the best rules maximizing the quality of mobile UI. Section 5 discusses the results of the evaluation of our approach. Finally, we conclude in Section 6.

2. Related works

Recently, there are several works that have focused on detecting defects of mobile user interface using different methods. Park et al.[20] has suggested an interview to assess the quality of mobile interfaces according to the user experience when using a mobile phone. Similarly, Gena and Torre[7] exploited a questionnaire to evaluate an MUI system called MastroCARONTE which provides tourist information on board cars. These methods are very expensive because they need the presence of many users having different profiles. For this reason, Soui et al.[24] have proposed a tracing system in order to evaluate the mobile application "MouverPerso".

In addition, one of the widely used methods to evaluate mobile interface is heuristic evaluation. In this way, Ji et al.[9] proposed a study that test usability checklist based on heuristic evaluations in views of mobile phone user interface practitioners. They reported also the importance of usability to evaluate the quality of mobile interface. Similarly, Biel et al[5] proposed a heuristic evaluation method based on generic evaluation scenario. Then, the results will be discussed and categorized according to their impact on usability and software architecture. These study propose manually heuristic to evaluate the mobile phone UI. In our work, we propose an automatic method to generate evaluation rules. In addition, Jin[10] proposed quantitative method to evaluate mobile UI in which they investigated the physical user interface. It measures the following three functions: key level (KLV), function level (FLV), grip level (GLV). Then, they estimate manually the degree of usability risk. Lettner and Holzmann[15] proposed also an semi-automatic usability evaluation method for mobile application UIs. It assesses MUI by using

log user interaction data and computing usability related metrics. Moreover, Miniukovich[17] applied the complexity based visual dimensions from the aesthetics model to study the popularity of mobile apps on Google Play. They proposed an automatic tool for MUI aesthetics evaluation called tLight which is based on visual aesthetic metrics to test if a MUI was visually simple and appealing, and which dimensions of MUI designs performed poorly. Kuparinen et al.[13] proposed also a set of heuristics for evaluating visual mobile map design. The introduced heuristics cover information about the user's location, route guidance, map scalability, adaptability of visible information depending on the device's screen size and application customizability to support user's personal interests.

This category of evaluation needs a lot of time to analyze and interpret the collected data and requires the explicit evaluator intervention. To this end, we propose generic detection rules for mobile interface evaluation. In fact, instead of inviting users whenever we have an interface to evaluate, the idea is to collect 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 application.

3. EVALUATION OF MUI

As pointed out by[20] user-centered evaluation can serve three goals: verifying the quality of a product, detecting problems and supporting decisions. According to[7], evaluation can be described as a process through which information about the usability of system is gathered in order to improve the system or to assess a completed user interface. The ergonomic evaluation of UI consists to verify whether the user is able to achieve his or her task using a given communication system[3, 19, 23, 25]. Senach[10] defines evaluation as follows "any evaluation needs to compare a model of the evaluated object to a referential model". In fact, during the ergonomic evaluation of the interactive system the observed model is then compared with the referential one. This model should be representative of the adequacy of the evaluated UI via the specific needs defined by the designer.

3.1. Metrics for MUI evaluation

Evaluation should be performed taking into consideration metrics and standards of evaluation. In this context, we propose a set of evaluation metrics inspired from[19] and based on several ergonomic criteria proposed by[4]. The values of these metrics are in [0, 1], where 0 present the lower value and 1 present the higher value. As shown in Table 1, we use eight metrics: Density, regularity, grouping, sequence and simplicity. (See Ngo et al.[19] for a more extensive description about metrics formulas).

Table 1. List of quality metrics

Metrics	Description
Density	It represents the percentage of workload of MUIs. It refers the extent to which interface is covered with objects (Components).
Regularity	It measures the uniformity of MUI components. It is sensitive to the number of vertical and horizontal alignment and to the spacing between components.
Sequence	It represents the extent to which the MUI is ordered based on reading pattern of left-to-right, top-to-bottom that is common in Western cultures.
Grouping	It is the number of group on the MUI. It is the degree to which all elements seem belong together.
Simplicity	It measures the complexity of MUI by counting the number of rows and columns on the interface. It is sensitive to the number of elements on the screen.
Homogeneity	It measures the distribution of elements among the four quadrants of the MUI. Well homogeneity is achieved by an equal distribution of the elements among the four quadrants of the MUI.

Unity	It measures the coherence of MUI elements. Unity is grouping all MUI elements to appear like a one piece.
Symmetry	It gives by an equal distribution of the quantity of elements on the right and the left columns of mobile user interface.

3.2. Visual Complexity Defects

In the literature, we can find a large list of MUI defects. In our study, we will focus on eight defects of MUI (see Table 2.). These defects are related in the evaluation of visual complexity of mobile user interface which is a crucial factor for user satisfaction.

Table 2. List of problems

Defects	**Description**
Incorrect layout of widgets	Incorrect layout of widgets: is related to the incorrect arrangement of MUI components (e.g., bad alignment, bad orientation, wrong position etc.)[14].
Overloaded	It is a bad density of MUI. In this case, the users find the MUI with high density and too crowded and not easy to read[6].
Complicated MUI	It is a measurement of structural sophistication of MUI layout. MUI includes more and more widgets to propose more features that are greater than the users' needs. So, this large number of features may lead to making MUIs complex[3].
Incorrect data presentation	It is an adaptiveness of the content which related to the incorrect extraction and presentation of information[22].
InCohesion of MUI :	It is a measure of the degree of interrelatedness of software component parts. In the MUI design, cohesion can be applied to show how the widgets are related to each other for a given MUI[2].
Difficult navigation	It is related to the lack of supplementary information and lack of descriptive labels that will be shown when a user selects an object (i.e. field). The latter problem is related to consistency [1].
Ineffective appearance of widgets	This problem occurs when MUI widgets follow an unexpected layout (e.g., wrong order). The presentation form and grouping of options does not match the mental model of the user so they react with confusion or cannot find the information although it is provided[6].
Imbalance of MUI	It is an unequal distribution of the quantity of interactive objects such as a button, a textfield and a textbox on the right and the left columns of a MUI.

4. Evaluation rules generation

Our contribution aims to define a set of detection rules as a combination of metrics quality, context criteria and a list of possible problem types (as described in figure 1.). This section shows how the generation of the evaluation rules can be seen as an optimization problem. We also show why the size of the corresponding search space makes meta-heuristic search necessary in order to explore it.

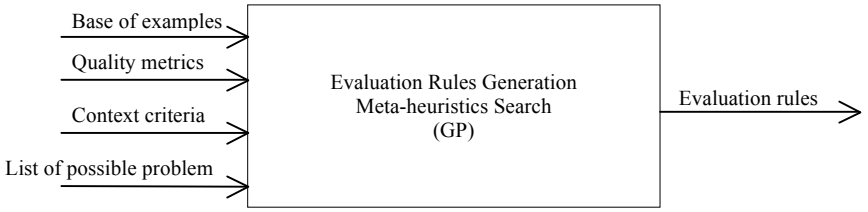


Fig. 1. The proposed approach overview.

The goal of our approach is to generate the best evaluation rules among all possible rules that minimize the visual complexity of MUI. The rule generation process aims to find the best combination between k context criteria, m quality metrics and p possible problems (defects). 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 possible 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: NR = (3mk). Considering these magnitudes, an exhaustive search cannot be used within a reasonable time frame. In such situations, or when a formal approach is not possible, the heuristic search can be used. The search is guided by the quality of the solution using the base of examples.

4.1. Genetic Programming Overview

Genetic programming (GP) is a powerful meta-heuristic search optimization method. It was proposed by[12] as a branch of Genetic Algorithm. The basic idea is to explore the search space by making a population of candidate solutions, also called individuals, evolve toward a "good" solution of a specific problem. GP is characterized by the representation of the solution as a tree. The tree composed on function set and terminal set, where the function set can be operators, function, or statements, and, the terminal set contains all inputs.

The choice of the genetic algorithm (GP) comes from the need to take advantage of the effectiveness of this principle in the living world where individuals are adapted to their environment. It is a method that has great flexibility because it offers the ability to handle a large number of individuals and to rework the solutions obtained by re-launching a new evolution starting with one or more solutions obtained previously. On the other hand, it is possible to keep more than one solution as a result of a development, to evaluate several over the basis of examples and so choose the one that offers the best fitness. This makes genetic programming an adaptable technique to our problem of generating evaluation rules. Indeed, the use of GP in our work results from the need to generate evaluation rules basing on examples that the best solutions produced along all generations. Each individual of the population is evaluated by a fitness function that determines its ability to solve the current problem. Then, it is subjected to the action of genetic operators such as reproduction and crossover. The reproduction operator selects individuals in the current population in proportion to their fitness values, so that the more fit an individual is, the higher probability that it will take part in the next generation of individuals. The crossover operator replaces a randomly selected part of an individual with a randomly chosen part from another individual to obtain one child. Finally, mutation operator is applied, with a probability which is usually inversely proportional to its fitness value, to modify some randomly selected nodes in a single individual. This process is repeated iteratively, usually for a fixed number of generations. The result of genetic algorithm is the best individuals found along all generations.

4.2. Genetic Programming Adaptation

By exploiting principle proposed in[11], the proposed approach based on using manually detected defects (trace). The idea is to use knowledge from previously manually evaluated mobile application (base of examples) in order to detect adaptation defects that will serve to generate new evaluation rules based on combinations of context criteria, quality metrics and defects. Thus, the evaluation rules are automatically derived by an optimization process that exploits the available examples. In this context, we use the Genetic programming (GP) that can be used to generate evaluation rules and detect defects of mobile user interfaces (taking account of the context of use and the quality metrics). The proposed method has four phases: (1) generation and evaluation of the initial population, (2) selection of parents, (3) generation of the population of children and (4) merge parents and children and evaluation of the new

population. The example base contains traces that were validated manually by an expert. The pseudo code for the algorithm is given in Figure 2.

```
Input:
  Set of context criteria, quality metrics and defects
  Trace.
Process:
  1: I:= Evaluation_rule (R)
  2: P:= set_of (I)
  3: initial_population (P, Max_size)
  4: repeat
  5: for all I in P do
  6: Detected_rule:= execute_rules(R)
  7: fitness (I):= C (Number of used metrics , Solution_size)
  8: end for
  9: best_solution := best_fitness(I);
  10: P := generate_new_population(P)
  11: it:=it+1;
  12: until it=max_it
  13: return best_solution
Output:
  best  solution: evaluation rule
```

Fig. 2. High-level pseudo-code for GP adaptation to our problem.

As Figure 2 shows, the algorithm takes as input a set of context criteria, a set of quality metrics and a set of defects. Lines 1–3 construct an initial GP population which is a set of evaluation rules. Lines 4–13 encode the main GP loop, which searches for the best evaluation rule. During each iteration, the quality of each individual is evaluated, and the solution having the best fitness is saved (line 9). Then, a new population of solutions is generated (line 10) using the crossover operator to the selected solutions; each pair of parent solutions produces two new solutions. We include the parents and child variants in the population and then apply the mutation operator to each variant; this produces the population for the next generation. The algorithm terminates when it achieve the maximum iteration number, and return the best set of evaluation rules.

Phase 1: Generation and evaluation of the initial population: this step takes in input context criteria (user experience, age, motivation, education level and motivation), quality metrics (density, grouping, regularity, sequence, simplicity, homogeneity, unity, symmetry), and possible mobile user interface defects (incorrect layout of widgets of MUI, overloaded MUI, complicated MUI, incorrect data presentation of MUI, difficult navigation, ineffective appearance of widgets of MUI, incohesion of MUI and imbalance of MUI). Note that each attribute of the context can take three values {high, medium, low} and the values of each metric are in [0, 1]. This step allows generating an initial population in a random manner from a possible combination of context criteria, quality metrics and defects. This generation is based on: 1) the correlation between the context criteria and the quality metrics. For example, if user has low experience, the mobile interface should have low density and high grouping (guidance). 2) the correlation between quality metrics and MUI defects, each metrics can be symptom for many problems. For instance, density is a symptom of overload MUI and also is a symptom of ineffective appearance of widgets of MUI, etc. This step will have as output a set of N randomly generated evaluation rules.

- **Encoding:** The GP needs a significant coding of individuals to facilitate the process of the algorithm. Indeed, the coding can have a significant impact on the way in which rules are processed. In our work, an individual is a set of declarative IF-THEN rules. The rule antecedent (IF part) describes the condition that combines some context criteria and quality metrics using the logic operators (AND). Each context criteria is a triple <attribute, operator, value>,

such as <motivation = low>, and each quality metrics is also a triple <attribute, operator, value>, such as <density >= 0.45>. The rule consequent (THEN part) highlights the problem types to be detected. This kind of rules representation has the advantage of being intuitively comprehensible for the user. An evaluation rule has the following structure:

IF "Combination of context criteria and quality metrics" THEN "type of detected problem"

The rules will undergo a process of improvement; they must be presented as a tree. For instance, let us consider the following example of evaluation rules:

R1: IF (motivation = low) **AND** (regularity <= 0.32) **THEN** Incorrect layout of widgets of MUI
R2: IF (user experience = medium) **AND** (grouping <= 0.17) **THEN** Difficult navigation
R3: IF (age = high) **AND** (simplicity <= 0.4) **THEN** complicated MUI

Tree-based modeling is one of the most exploratory techniques for representing rules trees. Thus, in this paper, we describe individuals using the representation of GP which can be viewed as a tree-based structure composed of two types of nodes: functions and terminals. The terminals (leaf nodes of a tree) correspond to different context criteria with their values and quality metrics with their threshold values. The function corresponds to the logical operators (AND) which used to relate context criteria and quality metrics. Figure 3 represents an example of evaluation rule which is represented as a tree. This tree representation corresponds to an OR composition of two sub-trees, each sub-tree represents a rule: R1 OR R2.

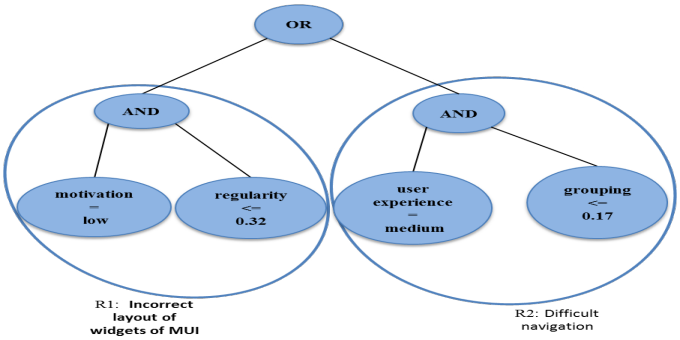


Fig.3. A tree representation of an individual.

- Evaluation of individuals of the initial population: In this step, we evaluate the generated solutions of the initial population (Pt) in order to estimate and discover the interesting evaluation rules. This evaluation is based on the calculation of the fitness function C(x) given in equation (1) which will evaluate the complexity of mobile user interface. It refers to determine the number of used metrics per solution. The idea is to improve the quality of evaluation rules by minimizing the function C(x). With xi is an individual. In fact, minimizing complexity of individuals means maximizing the number of detected defects. In fact, determine the complexity of MUI aims to improve the quality of MUI. Thus, low complexity of interface components provides an easy interface to use. Complexity is a measurement of structural sophistication of MUI layout. MUI includes more and more widgets to propose more features that are greater than the users' needs. So, this large number of features may lead to make MUIs complex. Our fitness function is calculated using the following formula:

$$C\left(xi\right) = \frac{\sum_{j=1}^{n} D_j + \sum_{j=1}^{n} G_j + \sum_{j=1}^{n} SM_j + \sum_{j=1}^{n} S_j + \sum_{j=1}^{n} R_j + \sum_{j=1}^{n} H_j + \sum_{j=1}^{n} U_j + \sum_{j=1}^{n} Sym_j}{SS} \quad (1)$$

Where:
D_i, SM_i, G_i, S_i, R_i , H_i, U_i, Sym_i= refers respectively to the number of density, simplicity, grouping, sequence, regularity, homogeneity, unity and symmetry in the solution xi.
{D_i, SM_i, G_i, S_i, R_i H_i, U_i, Sym_i}= 1 if the metric exist in the rule j.
{D_i, SM_i, G_i, S_i, R_i H_i, U_i, Sym_i } = 0 if the metric doesn't exist in the rule j.
SS: it is number of generated rule by solution (Size of Solution).

Phase 2: Selection of the parents
In this phase, two parents (individuals) are selected from the initial population. Thus, to guide the selection process, our method uses the fitness function to select elements having the best fitness from Pt that represents the best elements to be reproduced in the child population Qt using genetic operators such as mutation and crossover.

Phase 3: Generation of the children population
This phase is based on two principal genetic operators: crossover and mutation.
- Crossover: This operator consists of generating offspring. To this end, the proposed algorithm starts by choosing the crossover point (cut-point) that will be used to split the parents into two sub-tree. After that, the parents are combined to generate two new individuals (offspring). Each child combines information coming from both parents. In fact, this operator creates two children (solutions). The first sub-tree of the first parent to the second subtree of the second parent constructs the first child, and, the first sub-tree of the second parent with the second sub-tree of the first parent constructs the second child. Figure 4 shows an example of the crossover process.

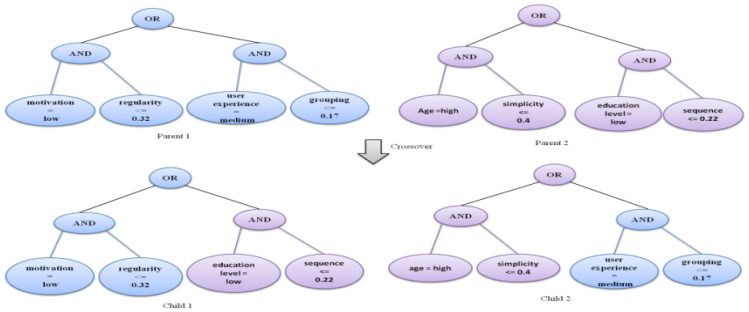


Fig. 4. Crossover operator.

- Mutation: after crossover has occurred, each child produced by the crossover undergoes mutation with a low probability. This phase consists of changing randomly one or more rules in the solution (vector). The modification of rule aims to change randomly the value of context criteria, the value of quality metrics, and the logical operator. Figure 5 shows a new solution.

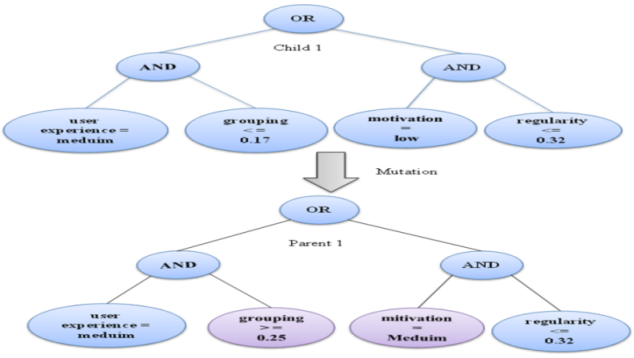


Fig. 5. Mutation operator.

Phase 4: Evaluation of the new population
Once reproduction and crossover have been applied according to given probabilities, parents and children are merged and the newly created generation of individuals is evaluated by the fitness function C(x). This process is repeated iteratively, usually for a fixed number of generations. The result of genetic programming (the best solution found) is the fittest individual produced along all generations.

5. VALIDATION

Our study addresses two main research questions, which are defined below. In this section, we explain how our experiments are designed to address them. The goal of the study is to evaluate the efficiency of our proposal for defects detection.

5.1. Research question

We designed our experiment to answer the following two research questions:
RQ1: How accurate is the proposed GP algorithm for defects detection?
RQ2: To what extent can the adaptation of multiple metrics impact on better investigation for defects in the mobile interface?

To assess the accuracy of our approach, we answer RQ1 by computing two measures, precision and recall that are originally defined in the domain of information retrieval and widely used when comparing search algorithms' results. As shown the equation (2), the precision denotes the fraction of correctly detected defects among the set of all detected defect:

$$\text{Precision} = \frac{\#\text{correctly reported defects}}{\#\text{number of detected detefcts}} \quad (2)$$

The recall (equation 3) indicates the fraction of correctly detected defects among the set of all manually identified by the base of example defects (that is, how many defects are undetected):

$$\text{Recall} = \frac{\#\text{correctly reported defects}}{\#\text{total number of defetcts in the base of example}} \quad (3)$$

In general, the precision calculates the probability that a detected defect is correct, and the recall is the probability that an expected defect is detected. Thus, both values are in between 0 and 1, and the higher value the better it is. To answer RQ2, we focus on the importance of using various metrics to achieve better quality of all the mobile user interfaces to evaluate.

5.2. Studied projects

The validation is being conducted over the evaluation of four known open source android applications: Duoling1, Accuweather2, Easy-loan-calculator3 (Simulation credit Pro), and Handicraft Women4. The corpus used includes releases of Duolingo which is a great application for learning languages. It customizes the course according to the user's goal (casual, regular, serious) and experience (beginner or medium). Accuweather is an application that provides prediction about the current and future weather. It has mobile user interface that changes depending on the location, time of day and weather conditions. This application allows users to personalize the number of displayed information according to their needs. Easy loan calculator is one of the best simulation credit applications that help users to simulate their personal loan. And, HandicraftWomen project which aims to support handcraft women in their business activities. It consists on adapting the current technologies to the profile of those handcraft women. We have chosen these projects because of their medium to large size; they considered as the most popular used application and can be used as input to our approach. They contain also multiple UIs using "relative" techniques which make the interfaces adjusted to the screen size.

5.3. Subjects

The evaluation of the mobile user interfaces of the studied project should improve the quality of his application and enhance the satisfaction of their users. As shown in Table 2, our work involved 20 subjects who have different age, motivation, level of education, experience and interest and using different mobile devices. All the subjects are volunteers and they have several profiles and preferences. In fact, 60% of participants were graduated (high education level), 15% have A-level (medium education level), and 25% have middle school level (low education level). The subjects having a high experience were 20%, 50% having medium experience and 30% having low experience. The data that were collected about the subjects show that 55% of participants reported having high motivation level, where 30% having medium.

Table 3. Subjects characteristics

Characteristics	**Measure**	**Subjects**	
		Number	**%**
Gender	Male	10	50
	Female	10	50
Age	Low: [18,30]	10	50
	Medium: [30,55]	4	20
	High: [55,80]	6	30
Education level	Low	5	25
	Medium	3	15
	High	12	60
Motivation	Low	3	15

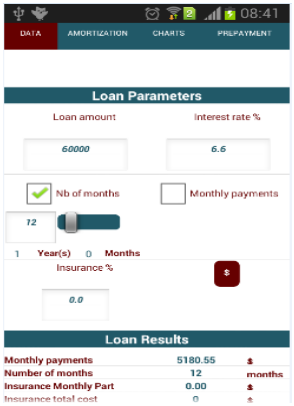
[1] https://github.com/KartikTalwar/Duolingo
[2] https://github.com/AccuWeather
[3] https://play.google.com/store/apps/details?id=appinventor.ai_Newbebiko.Simulation
[4] https://github.com/mabroukachouchane/HandcraftWomen/blob/master/FemmeArtisan.rar

	Medium	6	30
	High	11	55
Interest	Low	5	25
	Medium	6	30
	High	9	45
User experience	Low	6	30
	Medium	10	50
	High	4	20

5.4. Scenario

The subjects were invited to fill a questionnaire that aims to evaluate four mobile user interfaces of three studied mobile apps. This questionnaire was divided into two parts:

- User profile: contains the profile of the user who participates in the evaluation. The mobile user interfaces are evaluated with considering the context of use. So, the subjects were first asked to provide background information that includes the information in Table 2. According to[26], the profile of user is composed from five attributes that can take three values (Low, Medium, and High).
- User evaluation: After answering the first part, the subjects should test each mobile user interface by giving the defect type if it exists. These traces are validated by an expert on the field of evaluation of mobile apps.



MUI	Age	User Experience	Motivation	Education level	Interest	Defect
CreditData	Low	High	High	High	High	Incorrect layout of widgets

Fig. 6. Evaluation trace example of CreditData mobile user interface

5.5. Research question result

- **Result for research question 1**

As Figure 7 shows, the expected defects (defects reported in the trace) were detected with an average of more than 70% of precision on the five tested mobile apps. The highest precision for the applications was found in Easy loan calculator application where 90% of defects were detected. The lowest precision was found in Accuweather application with a median of 55% of detected defects. This is can confirm that the list of extracted defects did not contain a high number of false positives and so our reported defects can be trusted by the developers and they can include it in their manual inspections process.

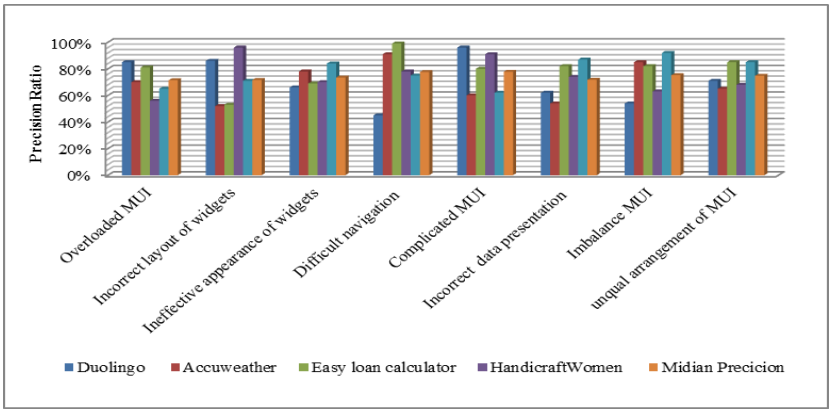


Fig. 7. GP precision of the defect detection solutions.

We found similar observation when analyzing the recall scores on the different mobile user interfaces where also an average of more than 70% of expected defects were detected. The highest and lowest median recalls ratios for the defects were respectively 98% for the complicated MUI and 45% for incorrect data presentation (see figure 8.). The recall scores of Duolingo application (largest application) are among the highest ones with more than 90% which are better than the detection results of the smaller application such as Easy loan calculator one.

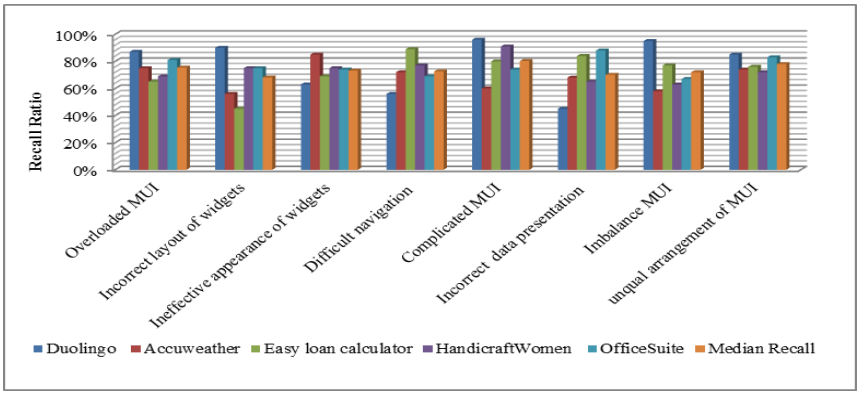


Fig.8. GP recall of the defect detection solutions.

- **Result for research question 2**

To study the impact of using multiple quality metrics, we run the process of generating rules and we examine the variation of fitness function values. The figure 9 shows that the fitness function increases when the number of used evaluation metrics increases also. In fact, increasing the number of metrics per solution means increasing the number of detected defects. This observation shows that using various metrics impact on the result of the evaluation process. They help the algorithm in better exploring the search space and generate near optimal solutions. Using different quality metrics allow the rules to capture more mobile user interface properties and quality defects, so, achieve better coverage of detected defects.

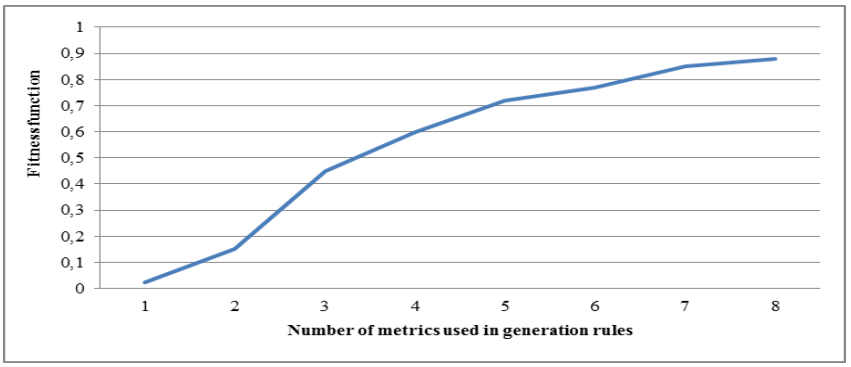


Fig. 9. Average Coverage Score of used metrics in solution.

6. CONCLUSION

Mobile user interface is a field in rapid evolution. The aim of such applications is to provide the user with relevant information that takes into account the context when using the system basing on the user preference learning. Today, these systems are indispensable to those who want to retrieve appropriate information with less effort at anytime and anywhere. The assessment of mobile user interface helps developers to minimize design defect. But, the evaluation of such systems is difficult and objects of very few propositions and studies in the literature. In this paper, we proposed to generate evaluation rules as an optimization problem using GP. These rules represent a combination of context criteria, quality metrics and quality defects. The obtained results confirm the efficiency of our technique with an average of more than 70% of precision and recall. However, we plane to continue our research for improving these numbers by using multi-objective algorithm.

References

1. Abrahão, S., E. Iborra and J. Vanderdonckt (2008). Usability evaluation of user interfaces generated with a model-driven architecture tool. *Maturing Usability, Springer: 3-32*
2. Alemerien, K. and K. Magel (2015). SLC: a visual cohesion metric to predict the usability of graphical user interfaces. *In Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM*.
3. Alemerien, K., & Magel, K. (2014). GUIEvaluator: A Metrictool for Evaluating the Complexity of Graphical User Interfaces. *In SEKE (pp. 13-18)*.
4. Bastien, J. C., & Scapin, D. L. (1993). Ergonomic criteria for the evaluation of human-computer interfaces *(Doctoral dissertation, Inria)*.
5. Biel, B., Grill, T., & Gruhn, V. (2010). 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.
6. Ehmke, C. and S. Wilson (2007). Identifying web usability problems from eye-tracking data. *In Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI* Volume 1, British Computer Society.
7. Gena, C., & Torre, I. (2010). The importance of adaptivity to provide onboard services: A preliminary evaluation of an adaptive tourist information service onboard vehicles. *Applied Artificial Intelligence*.
8. Hellmann, T. D., & Maurer, F. (2011, August). Rule-based exploratory testing of graphical user interfaces. *In Agile Conference (AGILE)*, 2011 (pp. 107-116).
9. Ji, Y. G., Park, J. H., Lee, C., & Yun, M. H. (2006). A usability checklist for the usability evaluation of mobile phone user interface. *International Journal of HumanComputer Interaction*, 20(3), 207-231.
10. Jin, B. S., & Ji, Y. G. (2010). Usability risk level evaluation for physical user interface of mobile phone. *Computers in Industry*, 61(4), 350-363.
11. Kessentini, M., Sahraoui, H., & Boukadoum, M. (2008). Model transformation as an optimization problem. *In Model Driven Engineering Languages and Systems* (pp. 159-173).
12. Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection (Vol. 1).

13. Kuparinen, L., Silvennoinen, J., & Isomäki, H. (2013). Introducing usability heuristics for mobile map applications. *In Proceedings of the 26th International Cartographic Conference* (ICC 2013).
14. Lelli, V., A. Blouin and B. Baudry (2015). Classifying and qualifying GUI defects. 2015 *IEEE 8th International Conference on Software Testing, Verification and Validation (ICST*), IEEE.
15. Lettner, F., & Holzmann, C. (2012, December). Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications. *In Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia* (pp. 118-127).
16. Memon, A. M., &Soffa, M. L. (2003). Regression testing of GUIs. *ACM SIGSOFT Software Engineering Notes*, 28(5), 118-127.
17. Miniukovich, A., 2016. Computational Aesthetics in HCI: Towards a Predictive Model of Graphical User Interface Aesthetics. Available at: http://eprints phd.biblio.unitn.it/1773/ [Accessed March 16, 2017].
18. Myers, B. A. (1995). User interface software tools.*ACM Transactions on Computer-Human Interaction (TOCHI*), 2(1), 64-103.
19. Ngo, D. C. L., Teo, L. S., & Byrne, J. G. (2000). Formalising guidelines for the design of screen layouts. Displays, 21(1), 3- 15.
20.Park, J., Han, S. H., Kim, H. K., Cho, Y., & Park, W. (2013).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.
21. Pascual, G. G., Pinto, M., & Fuentes, L. (2015). Selfadaptation of mobile systems driven by the common variability language. *Future Generation Computer Systems*, 47, 127-144.
22. Shitkova, M., Holler, J., Heide, T., Clever, N., & Becker, J. (2015, March). Towards Usability Guidelines for Mobile Websites and Applications. *In Wirtschafts informatik* (pp. 1603-1617).
23. Shneiderman, B., & Plaisant, C. (1994, August). The future of graphic user interfaces: Personal role managers. *In BCS HCI* (pp. 3-8).
24. Soui, M., Abed M., Kolski C., Ghédira K. (2012). Evaluation by simulation to optimise information systems' personalization quality in logistics, *International Journal of Production Research*, Volume 50, issue 13, pp. 3579-3593.
25. Vanderdonckt, J., & Gillo, X. (1994, June). Visual techniques for traditional and multimedia layouts. *In Proceedings of the workshop on Advanced visual interfaces* (pp. 95-104). ACM.
26. Yargin, G. T., & Crilly, N. (2015). Information and interaction requirements for software tools supporting analogical design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 29(02), 203-214.