Evolving Prototypes Towards The Best-suited Design and Interaction Schema Using The Genetic Algorithm

Ragaad AlTarawneh
Computer Graphics and HCI Group
University of Kaiserslautern
Kaiserslautern, Germany.
tarawneh@cs.uni-kl.de

Shah Rukh Humayoun
Computer Graphics and HCI Group
University of Kaiserslautern
Kaiserslautern, Germany.
humayoun@cs.uni-kl.de

ABSTRACT
The recent advances in the mobile environment, such as multi-touch gestures paradigm, introduce new challenges for the interaction designers in producing the best-suited final prototype. Moreover, the short delivery-time pressure of the current mobile market makes it harder to perform the detailed evaluations for selecting the best prototype amongst the created ones. In this vision paper, we propose an approach for evolving the created prototypes towards the final prototype with the *best-suited* design and interaction schema. Our approach is based on using the Genetic Algorithm for searching the *best solution (prototype with the best-suited design and interaction schema)* from the set of created prototypes during the design phase. The proposed approach suits the mobile application development and would enhance the interaction designers' ability of producing the final prototype of the target mobile application in an efficient and effective way.

Author Keywords
Interaction design, genetic algorithm, mobile environments.

ACM Classification Keywords
D.2.2 [Design Tools and Techniques]: Evolutionary prototyping, H.5.2 [User Interfaces]: Prototyping.

INTRODUCTION
Recently, we witness a large acceptance of using the mobile applications (commonly abbreviated as *mobile apps* or just *apps*) for performing different tasks, both in business processes and also in our daily life [3]. This is because of the simplicity of these mobile apps for specific tasks and the availability of mobile devices in most of our daily routine. The current single-task focusing paradigm [4] of these mobile apps makes them suitable for performing different activates – e.g., finding train timings, weather forecast, etc. – while in mobility. The impact of this is the users' growing requests for the availability of these mobile apps in short time. Due to this, software companies face the stress of launching their products in short time in order to fulfill the users' demands and also to compete with their market competitors.

Copyright is held by the owner/author(s).

PID-MAD 2013, Aug 27 2013, Munich, Germany.
(In Conjunction with *MobileHCI '13*, Aug 27-30 2013, Munich, Germany.)

The recent advances in the mobile interaction paradigm as well as the availability of a number of operating systems and mobile devices makes the designing process of these mobile apps an increasingly challenge for the interaction designers. During the design phase, interaction designers normally build a number of candidate prototypes, sometimes through involving end users in focus groups kind of meetings [3]. Evaluating and selecting the final prototype amongst these created ones is a time-consuming and efforts-taking process. Due to the short delivery-time pressure, normally the interaction design teams lack the time and resources for performing detailed evaluation in order to select the final prototype amongst the created ones. While even if the final prototype has been selected after the detailed evaluation, it is possible that few design elements or the attached interaction schema may not be the best-suited ones. It is also possible that the selected final prototype may provide better design and interaction for some parts while less for the remaining parts compared to the other prototypes. This may cause a revision in the design in later stages, which could make the development more costly and time consuming. Hence, selecting the final prototype with the best-suited design and interaction schema is a critical task for the current mobile application development world and plays an important role for the success of the end product.

To tackle this challenge, we propose an approach for evolving the created prototypes towards the final prototype while choosing the best-suited design and mobile interaction schema from all of the created prototypes through evolutionary steps. Our approach is based on using the Genetic Algorithm (GA) [1] for searching the *best solution (*prototype with the *best-suited design* and *interaction schema)* from the set of created prototypes. The genetic algorithm is based on search methods that employ processes found in natural biological evolution. These methods search or operate on a given population of potential solutions to find out a particular solution against some specification or criteria [2]. We propose to use the GA approach for generating the final prototype (*the best solution)* while checking against the required mobile app's functionalities, the design and interaction elements, and the target mobile environment. The *best solution* in our context means that the final prototype contains the *best-suited* design and interaction elements aiming at providing the users a better solution for performing their tasks in efficient and effective manners. The proposed approach would enhance not only the interaction designers' ability to produce the best-suited final prototype in an efficient and accurate way, but also decreases the time and the cost for reaching to this final prototype.

The remainder of this paper is structured as follows. In Section 2, we explain briefly the idea of the genetic algorithm. In Section 3, we introduce our approach for generating the best-suited final prototype through applying the genetic algorithm. We conclude in Section 4.

THE GENETIC ALGORITHM

The Genetic Algorithm (GA) [1] is a search technique that is used in computing to find true or approximate solutions in optimization and search problems. Genetic algorithm is categorized as global search heuristic [2]. The technique is implemented as a computer simulation in which a population of the abstract representations (called *chromosomes, genotypes,* or *genomes*) of the candidate solutions (called *individuals*, *creatures*, or *phenotypes*) for a combinatorial problem evolves toward the better solutions.

The evolution step usually starts from a population of randomly generated individuals. In each generation, the fitness of an individual in the population is evaluated; then multiple individuals are stochastically selected from the current population (based on their fitness), and finally modified (recombined and possibly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has been terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached. The genetic algorithm consists of the following four steps:

Chromosome Encoding

This process is responsible for representing the data into chromosomes. Each chromosome represents one of the candidate solutions in the search space. An example of a chromosome is shown in Figure 1, where it is represented by a set of float numbers. Each float number in this string represents some characteristic of the solution. There are plenty of ways for encoding the data [1, 2], such as through integers or real numbers, which mainly depends on the underlying problem.

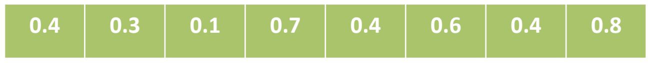

Figure 1. An encoded chromosome using the float numbers.

Crossover

In the crossover stage, the genes are selected from different parent chromosomes and then new offsprings have been created. The simplest way of doing crossover is to choose randomly some crossover point; where everything before this point is copied from the first parent while everything after this crossover point is copied from the second parent [1]. Figure 2 shows an example of creating two new offsprings from two parent chromosomes.

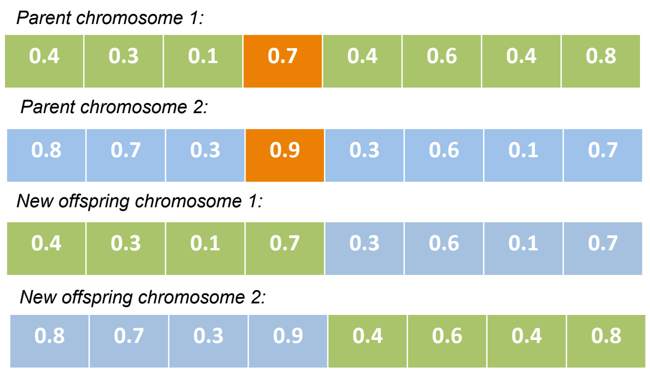

Figure 2. In this crossover example, the genes values in the orange boxes were selected to be the crossing points in order to generate the new offsprings.

Mutation

After the crossover has been performed, the mutation step is taken place. This is to prevent falling all solutions in population into a local optimum of solved problem [1, 2]. Mutation changes randomly the new offspring. Figure 3 shows the example of chromosome 1 (from Figure 1) after the mutation.

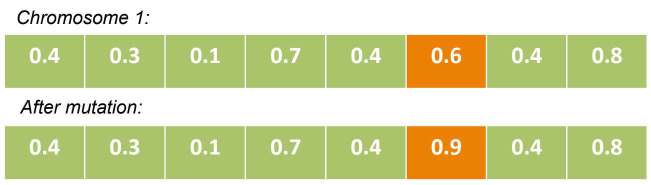

Figure 3. The mutation in the chromosome is done through swapping the value of one gene, as shown by the red color.

Elitism

In this stage, the best chromosomes (or the few best ones) are first copied and then are replaced with the old population in order to eliminate the bad chromosomes. The elitism process increases rapidly the performance of GA, because it prevents losing the best-found solutions.

The GA proceeds till the last three stages have not repeated to the maximum number of iterations or the GA reaches to the optimal solution.

THE METHODOLOGY

The goal of our approach is to evolve the created prototypes towards the final prototype that would deliver the users the possible *best-suited* design and mobile interaction schema in order to enable them to perform their tasks in efficient and effective ways. Our approach is based on reaching to the final prototype through applying the genetic algorithm approach in which the best solution (in our case that is the final prototype) is selected through an evolutionary process. This evolutionary process operates on a given population of potential solutions (in our case, these are the created prototypes by interaction designers in early stage) to find out a particular solution (the final prototype with the best-suited design and interaction schema) against some specification or criteria (e.g., UI elements, design layout, interaction elements and schema, target mobile environment, etc.).

We aim to find the best solution (i.e., the final prototype) with having the highest acceptance ratio of the design and the interaction schema. This acceptance ratio is measured using the weight value of the acceptance criteria, which in our case is a combination of the designed layout, the UI elements, the mobile interaction elements and schema, and the target mobile environment. We assume that the weight value of a particular functionality, required by the target mobile app, depends on how this functionality is formulated in the GUI. We also say that this weight value may differ from one prototype to another one for the same feature due to the different formulation of these combinational elements. These different variations between the weight values, due to the different formulation of combinational elements, define the fitness of the proposed solutions. In the forthcoming subsections, first we explain how to create chromosomes from the created prototypes in order to apply them in GA and then we introduce our approach complete workflow for applying it in the mobile application development.

The Chromosomes Creation Process

We assume that F is the set of functionalities – already elicited in the requirement phase – that is supposed to be provided by the target mobile app. Hence, the created prototypes should have provided the ways to achieve the required functionalities. While the $f_i \in F$ represents a single functionality in the set F. In the current mobile environment, not only the UI elements are important but also the design layout and the interaction schema (e.g. touch gestures) play important roles for the success of the end product. A functionality f_i can be formulated in different ways in different prototypes, depends on the combinational elements. For example, providing a zooming functionality to a frame area it is possible to formulate it in many ways. Either through a plus-and-minus button, or through a zooming in-out touch gesture with two figures, or through a combination of both. We assume that each required functionality f_i corresponds to a set M in which each element is a tuple (m, w). Each m, in the set M, represents a possible formulation (depends on the used UI elements, the design layout, the interaction elements and schema, and the target mobile environment) of the functionality f_i while the corresponding w represents the weight value of this formulation m through a float number ranging from 0 to 1. For example, in the above-described example we assume that the weight value of the plus-and-minus button formulation is 0.5, with the zooming in-out touch gesture formulation is 0.7, while with the both combination formulation is 0.9. The calculation of this weight value is based on many factors such as the user satisfaction level with a particular formulation, the target mobile devices and environment, the ergonomics facts about the formulation, etc. As this is out of the scope of this paper, so we assume that such standard studies would be done to give the weight values to different formulations.

The created prototypes by interaction designers are required to fulfill the functionalities set F. We assume that FP_n represents the set of functionalities provided by a particular prototype P_n, where n is the unique prototype number and $FP_n \subseteq F$. To create the chromosome c_n of a prototype P_n, we use the set FP_n from where we get the weight value of each functionality f_i by checking it in the corresponding set M against its formulation in the prototype P_n. So keeping the previous example, if the prototype p_1 provides the zooming functionality with the plus-minus button then the gene g_1 of the chromosome c_1 will have 0.5 weight value, while if the prototype p_2 provides the zooming functionality with zooming in-out touch gesture then the gene g_1 of chromosome c_2 will have 0.7 weight value. Through applying this technique for each functionality in the set F, we create a set of chromosomes C for the set P of all the created prototypes. These chromosomes will have the weight values of genes according to the corresponding functionality formulation in the created prototypes.

This set of chromosomes C, corresponding to the set P of the created prototypes, is used as the initial population in the GA. Then in the GA, a new set of the solutions is created using the GA operations (i.e., the cross-over, the mutation, and the elitism). The GA quits in two cases: the first one is when the result approaches to the optimal solution, while the second one is when the GA operations exceed to the maximum number of iterations. The result of these steps is a better solution by the time; i.e., a chromosome cf with a better fitness value survives. This resulting chromosome cf is then used to produce the *final prototype* that will have the best-suited design and interaction schema from the created prototypes.

The Workflow

Our proposed approach's workflow consists of six phases for evolving the created prototypes towards the best prototype. Figure 4 shows an overview of the approach's workflow. Following are the brief details of each phase:

- In the first phase, the software team defines the set of functionalities/requirements that are supposed to be provided by the target mobile application.
- Then the interaction design team creates a set of candidate prototypes for the target mobile application.
- If the set M has not been created earlier against the set of functionalities, then the team creates this set. This set M provides the weight value to a functionality based on how it is formulated in the prototype.

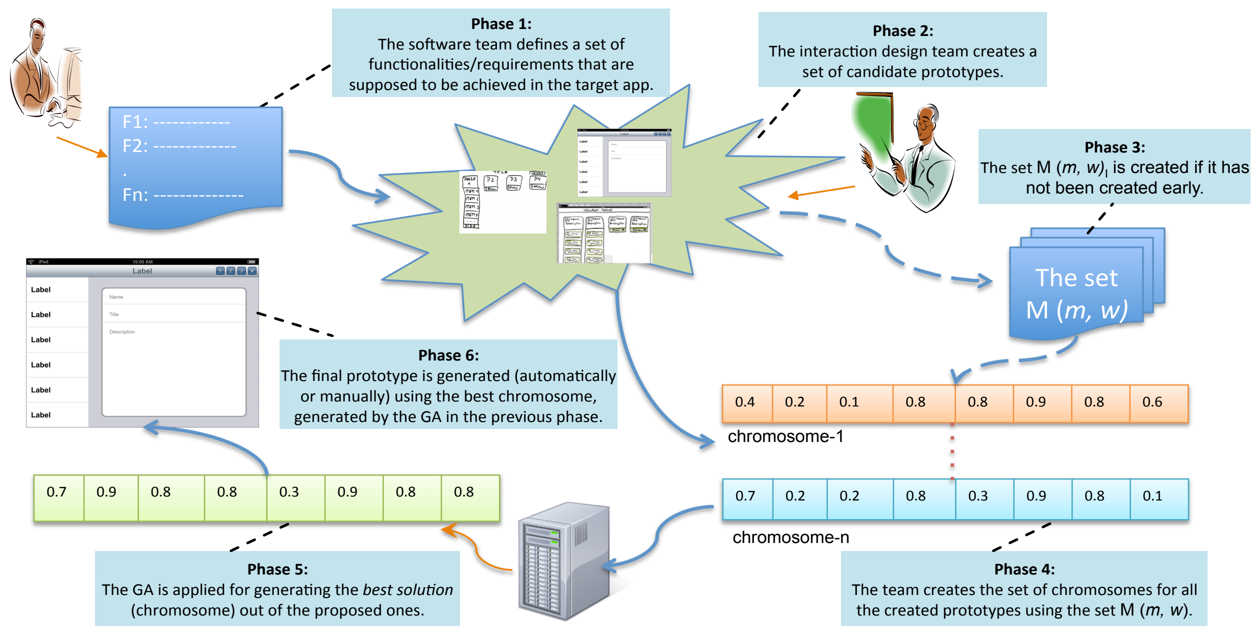
Phase 1:
The software team defines a set of functionalities/requirements that are supposed to be achieved in the target app.
Phase 2:
The interaction design team creates a set of candidate prototypes.
F1: ------------
F2: -------------
.
Fn: -------------
Phase 3:
The set M (m, w)ι is created if it has not been created early.
The set M (m, w)
0.4 0.2 0.1 0.8 0.8 0.9 0.8 0.6
chromosome-1
0.7 0.2 0.2 0.8 0.3 0.9 0.8 0.1
chromosome-n
Phase 6:
The final prototype is generated (automatically or manually) using the best chromosome, generated by the GA in the previous phase.
0.7 0.9 0.8 0.8 0.3 0.9 0.8 0.8
Phase 5:
The GA is applied for generating the best solution (chromosome) out of the proposed ones.
Phase 4:
The team creates the set of chromosomes for all the created prototypes using the set M (m, w).

Figure 4: The workflow for producing the *best-suited* prototype through applying the genetic algorithm.

- In phase four, the interaction design team creates the chromosomes against all the prototypes, created in phase two, using the set *M*. The genes in each chromosome will have the weight value according to how the functionalities are formulated in the underlying prototypes. The team calculates the final weight of each chromosome and if the summation of a chromosome is less than a specified threshold then the corresponding prototype is discarded from the solution. The remaining set of chromosomes will be the *input population* to the GA.
- In phase five, first the interaction design team defines the fitness function for the GA. The main constraint for the fitness function is to get the best chromosome, which will in fact represent the solution with the maximum weight value. The GA operations (i.e., the crossover, the mutation, and the elitism) are performed to generate more solutions (chromosomes) and the best solutions amongst them are then selected. The GA operations are repeated till they reach to the maximum number of iterations or the final chromosome converges to the optimal solution has been selected. This selected chromosome contains the maximum weight value.
- In phase six, the *final prototype* is generated against the best-generated solution (i.e., the final selected chromosome). The final prototype could be generated manually by cross checking the final selected chromosome's genes against the formulation in set *M* or through some automated tool support. The interaction design team could perform a quick evaluation on this generated prototype to see its effectiveness and efficiency.

CONCLUSION

In this paper, we proposed an approach for evolving the created prototypes towards the final prototype using the genetic algorithm. The approach searches the *best solution* (i.e., the prototype with the best-suited design and interaction schema) from the set of created prototypes. The approach is useful for reaching to the best solution accurately and in a cost-effective manner.

This is a first work towards applying the genetic algorithm in mobile app prototyping. Many things need to be done in order to utilize the approach with its full power. Especially, there is a need to do the studies for finding out the different combinational formulations of functionalities in prototyping and the weight value allocation to these formulations. In future, we plan to perform evaluation studies with mobile interaction design teams to check the feasibility and effectiveness of our approach.

REFERENCES

1. Banzhaf, W., Francone, F. D., Keller, R. E., and Nordin, P. Genetic Programming: an Introduction: On the Automatic Evolution of Computer Programs and its Applications. *Morgan Kaufmann Publishers Inc.*, San Francisco, CA, USA, 1998.
2. Levin, M. The Evolution of Understanding: A Genetic Algorithm Model of the Evolution of Communication. *Biosystems,* 36(3), 167-78, 1995.
3. Davis, R. C., Saponas, T. S., Shilman, M., and Landay, J. A. SketchWizard: Wizard of Oz Prototyping of Pen-Based User Interfaces. *UIST '07*, 119-128, ACM, 2007.
4. Lin, J., Newman, M. W., Hong, J. I., and Landay, J. A. Denim: Finding a Tighter Fit Between Tools and Practice for Web Site Design. *CHI '00*, 510-517, 2000.