SUPPORTING MENU LAYOUT DESIGN BY GENETIC PROGRAMMING

Cosimo Birtolo, Roberto Armenise
Poste Italiane S.p.A., Tecnologie dell'Informazione, Sviluppo Sistemi Informativi, Centro Ricerca e Sviluppo
Piazza Matteotti 3, 80133 Naples, Italy

Luigi Troiano
University of Sannio, CISELab, Department of Engineering, Viale Traiano, 82100 Benevento, Italy

Keywords: Menu layout, User interface, Genetic programming, Search based software engineering.

Abstract: Graphical User Interfaces heavily rely on menus to access application functionalities. Therefore designing properly menus poses relevant usability issues to face. Indeed, trading off between semantic preferences and usability makes this task not so easy to be performed. Meta-heuristics represent a promising approach in assisting designers to specify menu layouts. In this paper, we propose a preliminary experience in adopting Genetic Programming as a natural means for evolving a menu hierarchy towards optimal structure.

1 INTRODUCTION

Among the different components available to design a graphical user interface (GUI), menus are among the most prominent. Menu represents the major means by which the user gets access to the different application functionalities. Menu usability depends on several factors. Structure, layout, colors, labeling and control are among them. The issue regarding how to optimize menu usability by acting on those factors have been investigated in literature (Hollink et al., 2007; Cockburn et al., 2007; Zhang et al., 2007). Recently application of meta-heuristics to optimizing the layout of hierarchical menus have been proposed (Troiano et al., 2008; Matsui and Yamada, 2008). The problem of finding the layout that maximizes usability is combinatorial in nature, as it depends on the arrangement of each item in different positions onto the menu structure. This suggests that the problem is NP-hard. Genetic Algorithms (GA) have been applied with encouraging results (Troiano et al., 2008). Main limitation of this approach resides in the difficulty of modeling the evolution of hierarchical structures by means of linear chromosomes, as those processed by ordinary genetic algorithms. Attempts have been proposed in order to overcome this problem. For example Matsui and Yamada (Matsui and Yamada, 2008) encode menu layout as a sequence of menu items. Menu layout is built by assigning in turn menu items to nodes according to similarity of labels and load. Troiano, Birtolo, Armenise and Cirillo (Troiano et al., 2008) have proposed a coding in which each gene represents the path from root to a menu item. In some cases, an action can be accessed by different paths. For example, if an action is allowed twice, there is a need for two genes each representing a different path. The mapping between genes and actions is kept by an association table. Such a chromosome structure is more robust to genetic operations than others, allowing a better control of action items, whose best placement is the ultimate goal of the optimization algorithm. Main drawbacks to linear coding of menu layout are: (i) hierarchy imposes additional constraints to guarantee consistency of structure; (ii) a bigger number of non-valid solutions is produced; (iii) convergence becomes more difficult when additional structural and semantic constraints occur; (iv) a larger number of individuals and generations is required. Genetic Programming (GP) could overcome such drawbacks allowing the direct evolution of menu hierarchy.

In this paper we propose GP as an effective means for optimizing a menu layout as a trade-off between designer preferences and usability goals: Section 2 formulates the problem, Section 3 describes the algorithm, Section 4 provides preliminary experimental

Birtolo C., Armenise R. and Troiano L. (2010).
SUPPORTING MENU LAYOUT DESIGN BY GENETIC PROGRAMMING.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Human-Computer Interaction*, pages 248-251
DOI: 10.5220/0003022402480251

results, Section 5 outlines conclusions and future directions.

2 HIERARCHICAL MENUS

In this paper we will refer to *menu layout* as the hierarchical structure by which the user gains access to application functionalities. A menu layout is made of menus; each *menu* is made of a list of *items* referring to *submenus* or to *actions*. The firsts are menus at lower level, the latter are aimed at activating functionalities, thus they represent the menu system leaves (i.e. terminals).

In designing a menu layout we have to take into the account: (i) *accessibility* as the ease of reaching desired actions, (ii) *guidelines* as a set of best practices in organizing the menu layout, and (iii) *preferences*, as a wish list made explicit or implicit by the end user.

As the menu system aims at quickly activating functionalities, we will consider accessibility as the optimization driver, whilst we will refer to guidelines and preferences as optimization preferences (constraints, when mandatory). Therefore, the problem solution relies on finding a menu layout that maximizes accessibility and compliance to guidelines and user preferences. Menu selection entails user to visually inspect the menu, and to read and comprehend items in order to reach a desired functionality.

Several models for predicting the selection time have been proposed. If items are sorted, e.g. alphabetically, search time can be predicted by Hick's Law (Hick, 1952), which states that the time to locate an item is a logarithmic function of the menu size. When menus are not alphabetically ordered, users have to scan them in sequence the menu, but if the user memorizes the item position, search time becomes constant. Fitts' Law (Fitts, 1954; Cockburn et al., 2007) estimates the time required to move the cursor to a particular item, as a logarithmic function of the ratio between the target distance d and the target width w, known as the task's Index of Difficulty (ID).

Recently, Bernard (Bernard, 2002) has presented a further model for predicting the selection time. The Hypertext Accessibility Index measure (H_{HAI}) is defined as

$$H_{HAI}(x) = \sqrt{\sum_{i=1}^{L} \sum_{j \in N_i} \log_2(b_j + 1) \log_2(d_j + 1)} \quad (1)$$

where x is the menu structure, L the maximum number of levels of x, N_i the set of menus and menu items at level i, b_j the number of children of j, d_j the depth of j, assuming for the root and all menu item $d = 1$.

It can be easily verified that $H_{HAI} \in [1, +\infty)$: the lower H_{HAI} is, the more menu items are accessible; when all menu items are assigned to the root menu (i.e. no submenu is considered) $H_{HAI} = 1$. An interesting characteristic of this model is that H_{HAI} index predicts the expected navigation time on the basis only of the menu system layout. Bernard's model shows that though broader trees in general tend to have better search efficiency than deeper trees, topological shape has also an important effect. The H_{HAI} metric has been validated by comparing predictions with the empirical results found by others and Bernard himself.

Designers usually use guidelines to organize the menu structure. They provide a collection of best practices in organizing and structuring the menu layout. Examples are Apple's Human Interface Guidelines and Sun's Java Look and Feel Guidelines. Guidelines are either too specific or too vague, so they do not always apply to the problem at hand. For instance an Apple's Human Interface Guidelines suggests putting on menu bar some particular menus that an user expects to find such as "File", "View" and "Help". Guidelines say, as a general rule, to avoid creating long menus, in fact they are difficult for the user to scan and can be overwhelming, from other side it has not to put many items in a single menu and it needs to regrouping them in other menus. In most guidelines, it is suggested not to go further two levels of cascading menus, although in some cases it is convenient to violate this rule.

The importance of developing usable menu system is particular important in some domains such as mobile phone. Zhang et al.(Zhang et al., 2007) state menu displaying pattern should be identical with human cognition and is a key element for the efficiency of the communication. They prove that users, generally, preferred menu system in which the reaction time was faster and a hierarchical structure of menu could help users to improve small screen operating efficiency and their preference.

Hollink et al.(Hollink et al., 2007) address the optimization of menus with a purely navigational function and define the optimal menu as the one that minimizes the average time users need to reach their target pages. People are not always good at building hierarchies and at organizing menu items. However, building a quality menu system requires a large group of users (e.g. focus groups) and a large number of trials in order to find the best way or structuring the menu layout. Search techniques, can provide a valuable support in screening alternatives and in providing starting point that can be refined more efficiently and effectively.

3 ALGORITHM

The algorithm implemented a breeding sequence typical of evolutionary algorithms, that can be outlined as follows:

Algorithm 1. Algorithm structure.

```
Generate and evaluate a random population
repeat
  Select individuals for mating
  Cross selected individuals
  Mutate selected individuals
  Apply elitism
  Evaluate generated individuals
until generation limit is reached
```

Initial population is built using a procedure able to meet as much as possible the given constraints and preferences, thus assuring valid and highly fitted individuals since beginning. For selection, we adopted a tournament operator in order to reduce the effect of fitness scaling, since each individual is evaluated according to a fitness function defined on logical basis, as described below. Crossover and mutation have been implemented using a bid strategy, i.e. attempting to make a valid solution within a given number of trials. Elitism replaced random individuals with best individuals in order to improve performances, as this strategy does not require to sort the population before being applied. The algorithm was setup with standard parameters[1] as follows: Crossover 0.8, Mutation 0.02, Elitism 3.

In GP menu hierarchies can be represented directly by tree based chromosomes. Therefore each individuals represent one possible menu hierarchy. The fitness function of an individual x is aimed at modeling the trade-off between accessibility and preference compliance. Thus it is defined as convex combination

$$fitness(x) = \sigma \cdot H(x) + (1-\sigma) \cdot C(x) \qquad (2)$$

where $\sigma \in [0,1]$, $H(x)$ is the degree of accessibility, and $C(x)$ is the degree of preference compliance. In particular, $H(x)$ is defined as

$$H(x) = e^{k(1-H_{HAI}(x))} \qquad (3)$$

where $H_{HAI}(x)$ is defined by Eq.(1). The constant k controls the exponential decay.

[1]Parameter have been chosen by a simple qualitative analysis, according to common values adopted for them, without any in-depth quantitative analysis for their optimization.

Instead the degree of preference compliance is defined as the weighted mean

$$C(x) = \frac{\sum_{i=1}^{m} \bar{p_i} c_i(x)}{\sum_{i=1}^{m} \bar{p_i}} \qquad (4)$$

where m is the number of preferences, $\bar{p_i} = 1 - p_i$ is the constraint importance, and $c_i(x)$ is the compliance of x to the preference c_i. Therefore, we assumed a compensation between optimization criteria.

The problem of finding an optimal menu layout consists in placing all action items by maximizing accessibility and preference compliance. Preferences can be of different kinds. In our experimentation we considered the following types:

- **Path ordering** (*ancestor* , *successor*): defines a ordering relation between *ancestor* and *successor* along a path
- **Menu ordering** (*predecessor*, *follower*): defines a ordering relation between *predecessor* and *follower* whenever they coexist within the same menu
- **Number of menu items** (*menu*, *min*, *max*): defines the *min* and *max* number of items present in *menu*
- **Occurrence** (*item*, *min*, *max*): defines the *min* and *max* number of occurrences of *item*
- **Level** (*item*, *min*, *max*): defines the *min* and *max* level for *item*
- **Menu belonging** (*item*, *menu*): *item* should belong to *menu*

Each preference has a priority $p_i \in [1,5]$, where 1 is the highest priority (i.e. very important), 5 the lowest (i.e. not very important). The degree of compliance of x to each preference is computed as

$$c_i(x) = 1 - \frac{v_i(x)}{mv_i(x)} \qquad (5)$$

with $v_i(x)$ giving the number of criterion violations of x, and $mv_i(x)$ the maximum number of possible violations.

4 EXPERIMENTATION

As an example of application let us compose a menu layout made of 25 action items (terminals) and 12 submenus (non terminals). Experimentation was aimed at searching an optimal solution able to satisfy a set of 60 preferences with a different priority and

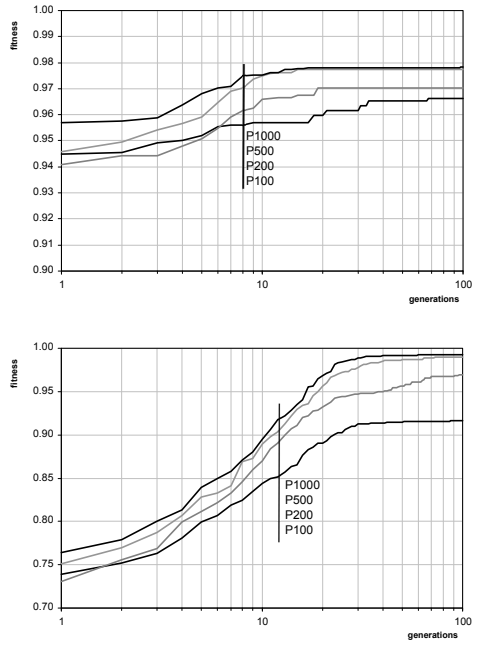

Figure 1: GP (top) and GA (bottom) fitness evolution.

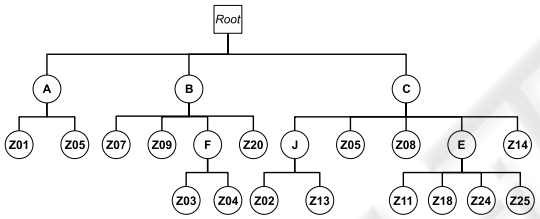

Figure 2: A possible solution given the set of preferences.

to score a good value of accessibility index defined in Eq. (1). Optimization was implemented using the framework Jenes (Troiano and De Pasquale, 2009), in order to make a direct comparison between GP and GA convergence as both algorithms are supported and implementation does not affect experimental results.

We run the algorithm 5 times with different population size (100, 200, 500 and 1000 individuals) in order to limit the effect of randomness in studying and comparing convergence. From Figure 1 we can outline some preliminary conclusions: (i) GP starts with better fitted individuals; (ii) GP convergence towards optimal solutions is faster in GP; (iii) population size is less relevant in GP than in GA. Figure 2 presents a possible solution.

5 CONCLUSIONS

In this paper we preliminarily presented a GP algorithm aimed at searching optimal trade-off between usability and semantic preferences. This approach seems to be promising and worth to be further investigated. In the future we wish to extend the approach presented here in order to find experimental evidence that solutions produced by machine are at least as good as those made by humans, at lower effort in time and resources, even considering a larger set of constraints. To reach this goal we plan to validate this approach by means of real application to Poste Italiane business case, and by blind test with focus groups where solutions provided by machine face solutions provided by humans.

ACKNOWLEDGEMENTS

Special thanks to Marco Merola (marco.merola@ciselab.org) for his technical contribution.

REFERENCES

Bernard, M. L. (2002). *Examining a metric for predicting the accessibility of information within hypertext structures*. PhD thesis, Wichita State University, Wichita, KS, USA. Adviser-Charles G. Halcomb.

Cockburn, A., Gutwin, C., and Greenberg, S. (2007). A predictive model of menu performance. In *CHI '07: Proc. of Int. Conf. on Human factors in computing systems*, pages 627–636, New York, NY, USA. ACM.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *J Exp Psychol*, 47(6):381–391.

Hick, W. E. (1952). On the rate of gain of information. *Quarterly J. of Experimental Psychology*, 4:11–26.

Hollink, V., Someren, M., and Wielinga, B. J. (2007). Navigation behavior models for link structure optimization. *User Modeling and User-Adapted Interaction*, 17(4):339–377.

Matsui, S. and Yamada, S. (2008). Genetic algorithm can optimize hierarchical menus. In *CHI '08: Proc. of Int. Conf. on Human factors in computing systems*, pages 1385–1388, New York, NY, USA. ACM.

Troiano, L., Birtolo, C., Armenise, R., and Cirillo, G. (2008). Optimization of menu layouts by means of genetic algorithms. In *EvoCOP*, volume 4972 of *LNCS*, pages 242–253. Springer.

Troiano, L. and De Pasquale, D. (2009). A java library for genetic algorithms addressing memory and time issues. In *Proc. of NaBIC 2009*, pages 642 –647. IEEE.

Zhang, X.-M., Shan, W., Xu, Q., Yang, B., and Zhang, Y.-F. (2007). An ergonomics study of menu-operation on mobile phone interface. In *Intelligent Information Technology Application, Workshop on*, pages 247 –251.