УДК 004.514

Построение меню при помощи алфавитного кода

И. В. Нечта[1]

В статье рассматриваются основные подходы к оптимизации интерфейса и существующие алгоритмы построения иерархического меню приложения. Автором предложен метод построения меню с плавающей шириной уровней иерархии, базирующийся на алфавитном коде Гилберта–Мура. Предложенный алгоритм имеет низкую трудоемкость и позволяет учитывать психофизиологические особенности человека при взаимодействии с компьютером.

Ключевые слова: оптимизация интерфейса, закон Хика, иерархическое меню.

1. Введение

При создании программного продукта большое внимание уделяется разработке пользовательского интерфейса. Считается, что при большом обилии конкурирующих программных продуктов, которые одинаково эффективно решают задачи пользователя, предпочтение будет отдано той программе, которая имеет наиболее легкое и удобное управление. Несмотря на то что к программе прилагается документация, достаточно полно описывающая порядок взаимодействия с программой, интерфейс стараются разработать интуитивно понятным для подавляющего числа пользователей. С одной стороны, даже неподготовленный пользователь должен быстро взаимодействовать с программой, как можно реже используя документацию; с другой стороны, интерфейс разрабатывается так, чтобы даже при длительном взаимодействии с программой внимание пользователя не притуплялось.

Проблема поиска алгоритма для построения оптимального интерфейса привлекает внимание многих зарубежных и отечественных исследователей. Данная проблема относится к междисциплинарным, и её решение может быть найдено на стыке двух наук: психологии и информатики.

Методы психологии позволяют изучить механизмы зрительного восприятия человека. Основные положения были изложены в работе [1], где определены принципы восприятия изображения человеческим глазом. В других работах, например [2], формулируются эргономические требования к элементам управления пользовательского интерфейса. Так, согласно закону «доминирования верхнего» считается, что самые важные элементы интерфейса следует размещать в правом верхнем углу. Указанные работы дают лишь рекомендации при построении интерфейса.

В отличие от психологии, методы теории информации позволяют однозначно рассчитывать время взаимодействия пользователя с программой при выполнении задачи. Так, скорость наведения курсора мыши и нажатия на элементы интерфейса может быть рассчитана по закону Фитса [4], устанавливающего логарифмическую зависимость между размером кнопки и временем наведения курсора на неё. Другие законы, например закон Хика, впервые предложенный в статье [5], позволяют оценить среднее время осуществления выбора одного из предложенных вариантов выбора. В общем случае эффективность интерфейса может быть оценена согласно модели GOMS [3], и, сравнивая различные версии интерфейса, разработчик

[1] Работа выполнена при частичной финансовой поддержке РФФИ, грант №15-29-07932.

выбирает наиболее приемлемый вариант. Однако такая модель не дает никаких рекомендаций относительно дальнейшей оптимизации.

Ряд публикаций, например [6–8], посвящен изучению механизмов работы мозга и особенностям реакций человека при осуществлении выбора. Так, считается, что закон Хика, являющийся аналогом энтропии в теории кодирования, не всегда может быть применен. Известны эксперименты, когда закон не находил своего подтверждения. Как было установлено позднее, результаты сильно зависят не только от предлагаемых вариантов выбора, но и от средств, которые пользователь задействует для ответной реакции (мышь, клавиатура, джойстик и т.д.). Указанные ограничения возникают вследствие особенностей алгоритмов принятия решений человеком. К сожалению, работа мозга в данный момент не до конца изучена, следовательно, при проведении эксперимента невозможно учесть все факторы, влияющие на результат. Тем не менее, наблюдается хорошая воспроизводимость результатов при одинаковых постановках эксперимента.

Одним из ключевых деталей интерфейса следует считать меню, где компактно размещаются основные элементы управления программой. Большое количество таких элементов вынуждает разработчиков группировать пункты меню. Соответственно обычное меню представляется в виде иерархической структуры, например, сильноветвящегося дерева. Множество работ посвящено разработке методов автоматического создания иерархии с минимальным среднем временем доступа к каждому элементу.

В данной работе предлагается метод автоматического построения меню, базирующегося на алфавитном коде Гилберта–Мура, применяемого для сжатия данных. Предложен алгоритм, позволяющий произвольно задавать ширину (количество элементов на одном уровне) иерархии.

2. Существующие подходы к оптимизации интерфейса

Ранее задача построения оптимального меню уже рассматривалась во многих работах, например в [9−11]. Были предложены оптимальные иерархические структуры с минимальным средним временем доступа к каждому элементу. Однако процесс размещения элементов по иерархии по-прежнему проводится в полуавтоматическом режиме. Основной проблемой, которая не дает полностью автоматизировать процесс, является невозможность группировки элементов в произвольном порядке. Это ограничение возникает у пунктов меню с сильно различающимся смыслом. Более того, разработчик принимает решение, какое наименование дать группирующим элементам меню. Таким образом, системы автоматического проектирования интерфейса ускоряют и упрощают процесс создания интерфейса и позволяют решать следующие задачи:

- Построение оптимальной иерархии при заданных вероятностях обращений к пунктам меню. В данном случае семантика не учитывается, что позволяет показать разработчику предел оптимизации по среднему времени доступа к элементам.
- Расчёт оценки качества меню. Оценка, полученная от двух вариантов меню, позволяет выбрать более приемлемый. Учитывая предыдущую задачу, разработчик, достаточно приблизившись к оптимальной структуре меню, может прекратить дальнейшую оптимизацию.
- Выявления «узких» мест интерфейса. Системы автоматического проектирования позволяют выявить наиболее важные места интерфейса, с которым пользователь будет чаще всего работать. Именно эти места разработчику необходимо оптимизировать в первую очередь.
- Генерация предварительного варианта меню.

Рассмотрим более подробно аспекты, возникающие при автоматическом построении меню. В зависимости от пользовательской стратегии поиска информации необходимо выбирать

оптимальную структуру, позволяющую достичь минимальное среднее время нахождения элемента. В работе [9] указывались две стратегии взаимодействия пользователя с меню:

- исчерпывающий поиск – пользователь полностью просматривает все элементы меню и затем принимает решение о выборе наиболее подходящего для него;
- последовательный поиск – пользователь последовательно просматривает варианты до тех пор, пока не встретит нужный.

Также утверждалась, что оптимальной структурой меню для исчерпывающего поиска следует считать однородное дерево, в котором каждый уровень иерархии имеет одинаковый размер и одинаковую суммарную вероятность. Для последовательного поиска более подходящим является неоднородная структура, в которой самые популярные элементы выносятся вверх иерархии.

Другой аспект связан со смысловой схожестью сгруппированных элементов. Пользователь, придерживающийся стратегии последовательного поиска, перемещаясь вглубь иерархии, должен однозначно определять путь перехода на каждом шаге, иначе ему потребуется возвращаться вверх всей иерархии или менять стратегию поиска, что значительно увеличит время работы с меню. В работе [15] оценка эффективности интерфейса представлена в виде трех слагаемых: среднего времени обращения к элементам меню и двух штрафных функций. Штрафные баллы начисляются за увеличение глубины иерархии и группировку сильно различающихся по смыслу элементов. Смысловая схожесть пунктов меню определяется с помощью функции-метрики. Метрика может быть задана в автоматическом режиме. Например, смысловая схожесть элементов меню тем выше, чем больше совпадающих слов в описании функции пункта меню по документации. Другой неавтоматизированный способ задания метрики предполагает составление таблицы, например табл. 1, в которой отмечается принадлежность элементов меню к смысловым категориям.

Таблица 1. Сопоставление элементов семантическим группам

Элемент меню	Семантическая группа				
	Персонализация	Сеть	Оформление текста	Настройки	Браузер
Дата и время		+		+	
Установки шрифта	+		+	+	
Свойства подключения	+	+		+	+

Чем больше совпадающих групп, тем ближе по смыслу находятся элементы. Таким образом, используя данную таблицу можно в автоматическом режиме производить построение предварительного меню. Более того, смысловые группы иногда могут выступать в качестве названий группирующих меню. В статье [15] подчеркивалось, что задача полного перебора всех вариантов меню, с разной шириной уровней, является NP-сложной и уже при размере меню, состоящем из 200 элементов, и максимальной ширине иерархии в 10 элементов трудоемкость составляет $O(n) \sim 10^{243}$.

Еще один возникающий аспект – это предельная ширина уровня иерархии. Согласно закону Хика, чем больше элементов, тем дольше будет пользователь осуществлять выбор, но, с другой стороны, на переход между уровнями меню также затрачивается время. Оптимальным количеством считается количество от 5 до 8 элементов. Однако иногда это количество может быть увеличено через зрительную группировку объектов, как например, на рис. 1, где часть элементов отделяется разделительной полосой.

В то же время в работе [12] указывается увеличение времени взаимодействия с расширяемым меню, если предыдущие пункты не были удалены с экрана. Следовательно, зрительная группировка не всегда способствует оптимизации.

Ряд работ, например [13–14], посвящен проблеме нахождения баланса между глубиной и шириной иерархии. Указывается, в ходе экспериментов достигнута высокая скорость в 64-х элементном меню при иерархиях [4,16][2] и [8,8]. Другие варианты иерархий, например [16,4], не давали увеличения скорости.

В данной работе предлагается метод автоматического построения меню, базирующийся на алфавитном коде Гилберта–Мура. Предложенные модификации алгоритма построения кода позволяют задавать произвольную ширину иерархии меню.

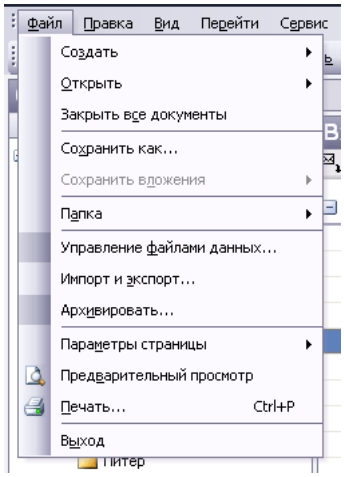

Рис.1. Группировка пунктов меню с помощью разделительной полосы

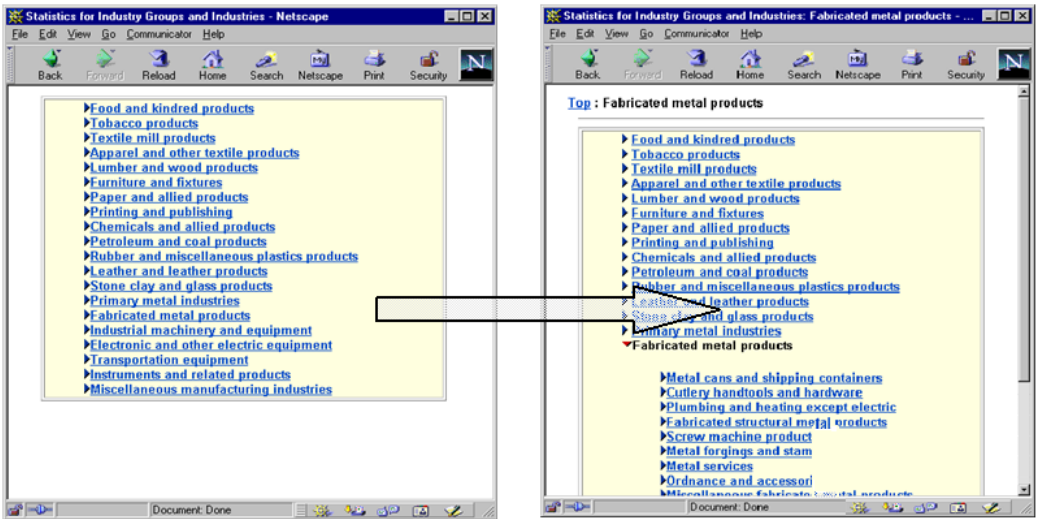

Рис. 2. Расширяемое меню

3. Описание предлагаемого подхода

Построение меню с использованием алфавитной группировки уже ранее предлагалось в работе [16]. Математическое моделирование позволило построить код и, соответственно, иерархию меню, близкую к оптимальному. В качестве моделируемого распределения вероятностей использовалось распределение Ципфа. Однако применение указанных алгоритмов не гарантирует оптимальность для произвольных частот вызова пунктов меню. Напротив, использование кода Гилберта–Мура благодаря своим особенностям позволяет нам достичь малой избыточности при абсолютно любых исходных частотах:

[2] Здесь указана ширина уровней иерархии (первый уровень – 4 пункта меню, второй - 16 пунктов).

$$L_{\text{ср}} < H + 2, \quad (1)$$

где $L_{\text{ср}}$ – средняя длина кодового слова (глубина пункта меню в иерархии), H – информационная энтропия (показывает предел оптимизации иерархии).

Рассмотрим на примере процесс построения иерархии с произвольной шириной уровня меню. Пусть имеется меню, состоящее из множества городов ($City$), задано распределение вероятностей их выбора (P) и задано множество ширин уровней ($W = \{3,2,3,2\}$), как показано в табл. 2. Множество $City$ упорядочено в лексикографическом порядке.

Таблица 2. Построение кода Гилберта–Мура

City	P	Q	L	Обычный код Г.-М.	Модифицированный код[3]
Барнаул	0.2	0.1	4	0001	0011
Бийск	0.1	0.25	5	01000	0111
Минск	0.5	0.55	2	10	1101
Новгород	0.1	0.85	5	11011	2100
Ростов	0.1	0.95	5	11110	2120

Процесс получения модифицированного кода производится следующим образом. Обычный код получается путем перевода кумулятивной вероятности Q в двоичную систему счисления, причем кодовое слово i составляют из $L_i = \lceil -\log_2 p_i \rceil + 1$ двоичных знаков после запятой. При переводе дробного числа в двоичную систему счисления производится итеративное умножение дробной части на коэффициент, равный основанию новой системы счисления (т.е. 2). Целые части чисел, полученных при умножении, составляют переведенное число. Главное отличие модифицированного кода заключается в том, что умножение дробной части производится на элементы из множества W. В табл. 3 представлен процесс перевода числа 0.95 для $W = \{3,2,3,2\}$.

Таблица 3. Процесс получения модифицированного кода

Обычный процесс перевода числа в двоичную систему счисления			Процесс получения модифицируемого кода		
Целая часть	Дробная часть	Помножаемый коэффициент	Целая часть	Дробная часть	Помножаемый коэффициент
0	95	2	0	95	3
1	9	2	**2**	85	2
1	8	2	**1**	7	3
1	6	2	**2**	1	2
0.95→0.111…			0.95→0.212…		

Еще один важный вопрос, который возникает при построении кода – сколько знаков после запятой оставлять в кодовом слове. Здесь следует руководствоваться следующей схемой:

1. Изначально оставляем $S = \max(L_i)$ знаков после запятой, где L_i – двоичные кодовые слова.
2. Разобьем на группы кодовые слова с совпадающими знаками на позиции $I = 1$ после запятой. В нашем примере получится 3 группы: ({0.**0**011, 0.**0**111} {0.**1**101} {0.**2**100, 0.**2**120}).
3. Если в группе остался только один элемент, то все знаки после текущей позиции отбрасываем – (0.**1**~~101~~) и дальнейшая обработка такой группы завершается.
4. Разбиваем аналогично каждую группу на подгруппы при $I = I + 1$.

[3] Здесь для упрощения показана точность до 4 знаков после запятой.

5. Пока не будет завершена обработка всех групп, переходим на шаг 3.

Таким образом, получится следующий код:

Таблица 3. Итоговый код

Город	Модифицированный код	Итоговый код
Барнаул	0011	00
Бийск	0111	01
Минск	1101	1
Новгород	2100	210
Ростов	2120	212

В результате мы строим дерево, соответствующее коду, как показано на рис. 3. Как видно из рисунка, у нас имеется избыточный узел в последних двух кодовых словах. С точки зрения оптимального построения кода мы могли бы в кодах, соответствующих «Novgorod» и «Rostov», убрать второй знак (21~~1~~0, 21~~1~~2). Однако в общем случае такая оптимизация может приводить к нарушению условий, накладываемых на размер ширины уровней иерархии.

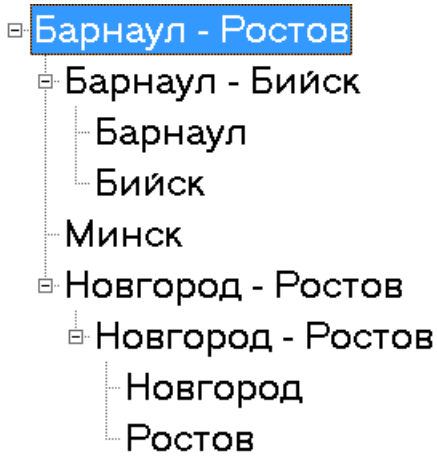

Рис 3. Построенное меню

Таким образом, был разработан алгоритм и программное средство, позволяющее производить построение иерархического меню, близкое к оптимальному по среднему времени поиска элемента. Алгоритм имеет низкую трудоемкость и позволяет задавать различные размеры ширины иерархии на каждом уровне.

Литература

1. *Gibson J. J.* The ecological approach to visual perception. – 1986.
2. *Рыжов В. А., Корниенко А. В., Демидович Д. В.* Качество экранных изображений в обучающих программах // Педагогическая информатика. – 2002. – Т. 1. – С. 42–55.
3. *Card S. K., et al.* The psychology of human-computer interaction. – 1983.
4. *Fitts P. M.* "The information capacity of the human motor system in controlling the amplitude of movement". Journal of Experimental Psychology 47 (6). (June 1954). P. 381–391. doi:10.1037/h0055392. PMID 13174710.
5. *Hick W. E.* On the rate of gain of information // Quarterly Journal of Experimental Psychology, 1952. V. 4. P. 11–26.
6. *Dassonville P, et al.* Choice and stimulus–response compatibility affect duration of response selection // Cognitive Brain Research. 1999. V. 7. P. 235–240.

7. *Wright C. E., et al.* Visually guided, aimed movements can be unaffected by stimulus–response uncertainty // Experimental Brain Research. 2007. V. 179. P. 475–496.
8. *Leonard J. A.* Tactual choice reactions: I. Quarterly Journal of Experimental Psychology. 1959. V. 11. P. 76–83.
9. *Губко М. В., Даниленко А. И.* Математическая модель оптимизации структуры иерархического меню // Проблемы управления. 2010. №4. С. 49–58.
10. *Witten I. H., Cleary J. G., Greenberg S.* On frequency-based menu-splitting algorithms // International Journal of Man-Machine Studies. 1984. V. 21, №. 2. P. 135–148.
11. *Aaltonen A., et al.* 101 spots, or how do users read menus? // Proceedings of the SIGCHI conference on Human factors in computing systems. 1998. P. 132–139.
12. *Zaphiris P., et al.* Expandable indexes versus sequential menus for searching hierarchies on the world wide web, 15–99 (1999) [Электронный ресурс]. URL: `http://www.cs.umd.edu/hcil/pubs/tech-reports.shtml` (дата обращения: 06.06.2015).
13. *Kiger J. I.* The depth/breadth trade-off in the design of menu-driven user interfaces // International Journal of Man-Machine Studies. 1984. V. 20, №. 2. P. 201–213.
14. *Zaphiris P. G.* Depth vs Breath in the Arrangement of Web Links // Proceedings of the Human Factors and Ergonomics Society Annual Meeting. 2000. V. 44, №. 4. P. 453–456.
15. *Yamada M., et al.* Optimizing hierarchical menus by genetic algorithm and simulated annealing. // Proceedings of the 10th annual conference on Genetic and evolutionary computation. 2008.
16. *Witten I. H., et al* On frequency-based menu-splitting algorithms // International Journal of Man-Machine Studies. 1984. V. 21, №. 2. P. 135–148.

Статья поступила в редакцию 22.06.2015

Нечта Иван Васильевич
к.т.н., доцент кафедры прикладной математики и кибернетики СибГУТИ, начальник отдела подготовки кадров высшей квалификации СибГУТИ (630102, Новосибирск, ул. Кирова, 86) тел. (383) 2-698-358, e-mail: `www@inbox.ru`

Using Alphabet code for menu construction

I. Nechta

In this article, we consider main approaches to interface optimization and existing algorithms for constructing application hierarchical menu. A method for constructing a menu with floating width of hierarchy levels, based on the alphabetical Gilbert-Moore code is suggested by the author. The proposed algorithm has low labor- intensiveness and allows us to take into account physiological characteristics of human – computer interaction.

Keywords: interface optimization, Hick Law, hierarchical menu.