*ПРОГРАММИРОВАНИЕ, 2020, № 6, с. 30–40*

## ПРОГРАММНАЯ ИНЖЕНЕРИЯ, ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММ

*УДК 004.42*

# АЛГОРИТМ АДАПТИВНОГО ИЗМЕНЕНИЯ МЕНЮ ПРОГРАММНОГО ПРИЛОЖЕНИЯ

**© 2020 г. Е. Н. Чуйкова[a],*, А. Р. Айдинян[a],**, О. Л. Цветкова[a],*****

[a] *Донской государственный технический университет, 344002 Ростов-на-Дону, пл. Гагарина, 1, Россия*

**E-mail: elenchu@mail.ru*

***E-mail: andstyle@mail.ru*

****E-mail: olga_cvetkova@mail.ru*

Поступила в редакцию 24.02.2020 г.
После доработки 17.04.2020 г.
Принята к публикации 01.06.2020 г.

Меню программного приложения представляет собой важную часть его интерфейса для взаимодействия с пользователем. Удовлетворенность пользователя и эффективность использования программного приложения зависит от того, насколько правильно построено меню. Спроектированные разработчиком меню ориентированы на среднестатистического пользователя и не учитывают потребности и особенности отдельных пользователей. Решением данной проблемы является построение адаптивных пользовательских интерфейсов, обладающих способностью подстройки под нужды конкретного пользователя. Предложенный в статье подход к адаптации пользовательского меню программного приложения выполняется на основе модели пользователя, формируемой в результате наблюдения за действиями пользователя в процессе работы с программой. Построены математическая модель меню программного приложения и модель пользователя программного приложения, предложен алгоритм адаптивного изменения меню программного приложения. Благодаря предложенному алгоритму создания ссылок на пункты меню на более высоком уровне и скрытия неиспользуемых пунктов меню удалось уменьшить время активации пунктов меню, что повышает эффективность работы пользователей программного приложения.

**DOI:** 10.31857/S0132347420060035

## 1. ВВЕДЕНИЕ

Программный интерфейс, приспособленный к потребностям конкретных пользователей, является важным средством обеспечения эффективности применения программного приложения [1]. Поэтому усилия многих разработчиков направлены на решение данной задачи. В [2] представлена реализация концепции поддержки пользователей в контекстно-зависимом мобильном устройстве с адаптивным пользовательским интерфейсом. В [3] описан подход на основе шаблонов проектирования к реализации адаптивных пользовательских интерфейсов для пользователей с особыми потребностями. В [4] представлена система автоматической адаптации пользовательских интерфейсов, использующая модель идентификации анонимных пользователей программных продуктов и динамический идентификатор для автоматической адаптации интерфейса к потребностям идентифицированного пользователя. В [5] предложена архитектура адаптивной интерактивной системы для приложений из области электронного бизнеса, способной оценивать текущую ситуацию и реагировать на изменение контекста, среды и эмоций пользователя. Система содержит компонент, осуществляющий наблюдение за поведением пользователя в процессе его взаимодействия с системой. Необходимые модификации программного обеспечения выполняются в режиме реального времени. В [6] для решения задачи построения адаптивного пользовательского интерфейса предложена система анализа взаимодействий, которая с помощью вероятностных методов прогнозирует взаимодействия с пользователем, распознает действия пользователя и определяет

его предпочтения на разных уровнях абстракции. В [7] представлен анализ текущих достижений в области адаптивных пользовательских интерфейсов, определены перспективные направления в этой области, предложена таксономия для сравнения различных систем адаптивных пользовательских интерфейсов, выявлены общие принципы их эффективного проектирования, такие как персонализация методов взаимодействия с пользователей на основе учета его предпочтений, фильтрация нерелевантной информации с целью снижения когнитивной перегрузки пользователя, информационное сопровождение пользователя в процессе освоения нового приложения, включая обнаружение и исправление неправильных действий, объяснение новых функций и предоставление информации для упрощения решения задач.

Неотъемлемой составной частью любого пользовательского интерфейса является меню, поэтому построение адаптивного пользовательского интерфейса невозможно без обеспечения возможности оперативной перестройки его меню. Меню современного программного приложения отличается сложной древовидной структурой, в которой пункты меню находятся на разных уровнях. Чем ниже в иерархии расположен пункт меню, тем больше времени требуется на доступ к нему. Также на время доступа к элементу меню влияет количество пунктов в подменю, их последовательность, запоминаемость меню, опыт пользователя [8]. Усилия многих разработчиков были направлены на создание дружественного меню, интуитивно понятного любому пользователю [9]. Однако такое меню ориентировано на среднестатистического пользователя и не учитывает потребности отдельных пользователей. Решением данной проблемы является построение пользовательских меню, обладающих способностью подстройки под нужды конкретного пользователя [10, 11].

В [12, 13] определены четыре вида меню по способу адаптации:

– адаптируемые пользователем;

– адаптируемые пользователем с поддержкой системы;

– адаптивные – весь процесс адаптации управляется системой;

– адаптивные с контролем пользователя – система выполняет адаптацию под наблюдением пользователя.

Таким образом, выделяют адаптивные и адаптируемые меню, отличающиеся тем, что адаптируемые меню пользователь может сам приспосабливать к своим предпочтениям (например, путем выбора параметров, определяющих внешний вид пользовательского интерфейса).

При использовании адаптируемых меню в программном приложении необходимо предусмотреть соответствующие инструментальные средства, позволяющие пользователю изменять меню, добавлять панели инструментов, назначать действия кнопкам панелей инструментов и т.п. [14, 15]. Такие изменения в меню способен внести опытный пользователь, хорошо знающий систему и средства ее модификации.

При автоматической настройке меню программного приложения модифицируется без участия пользователя на основании имеющейся о нем информации (модели пользователя). При этом возможны негативная реакция пользователя и ошибки в предсказании действий пользователя. Поэтому адаптационные изменения должны проводиться с учетом этих факторов [5, 16–18].

Меню определяет набор возможных действий пользователя с объектами программного приложения. Адаптивность меню выражается в изменении этого набора в зависимости от потребностей определенного пользователя. Например, если имеются редко используемые пользователем пункты меню, то меню изменяется путем сокрытия этих элементов.

В ряде работ представлены результаты сравнения производительности адаптивного, адаптируемого и статического меню, приводящие к неоднозначным выводам. Так, в [19] показано, что большинство пользователей отдают предпочтение статическим меню, а в работе [20] утверждается, что адаптируемые меню являются более предпочтительными по сравнению с адаптивными меню, которые, в свою очередь, не хуже статических. Наиболее эффективным является сочетание автоматически адаптируемых меню с элементами ручной настройки [15, 21]. Причем ручная настройка должна выполняться удобным для пользователя образом без входа в специальные режимы настройки меню. Для ускорения доступа к элементам меню выполняется перенесение часто используемых пунктов в верхнюю часть (разделенные меню), выделение часто используемых пунктов (шрифтом, цветом) без перенесения из привычного места [8].

В [22] предложен метод оптимизации иерархических меню, позволяющий определить наиболее подходящее распределение элементов в древовидной структуре меню. Оптимизация выполняется на основе эвристических алгоритмов (имитация отжига и генетические алгоритмы). Однако использование меню с изменяющейся структурой

может снизить производительность работы пользователя вследствие нестабильности меню и дополнительных затрат времени, требуемых пользователю на изучение незнакомого меню [22], поэтому более целесообразно предусмотреть средства ускорения доступа к часто используемым элементам меню без существенного изменения его структуры посредством перераспределения и переупорядочивания элементов меню. Эта идея реализована в предложенном в данной статье алгоритме, обеспечивающем динамическую адаптацию меню посредством создания ссылок на часто используемые пункты меню и добавления этих ссылок в меню вышележащего уровня. По мере активации ссылок создаются ссылки на еще более высоком уровне. Кроме осуществляемой таким образом автоматической адаптации меню предусмотрены средства ручной настройки, предоставляемые пользователю.

## 2. МАТЕМАТИЧЕСКАЯ МОДЕЛЬ МЕНЮ ПРОГРАММНОГО ПРИЛОЖЕНИЯ

Меню программного приложения характеризуется навигационной структурой, представляющей собой иерархию пунктов меню. Все пункты меню можно разделить на две группы:

1) пункты меню, активирующие команду;

2) пункты меню, содержащие подменю.

Иерархию элементов меню можно представить в виде $N$-арного дерева – связного ациклического графа. Корень дерева (root) соответствует меню программы. По определению каждый узел дерева, кроме корня, может иметь только одного предка. Узел дерева может порождать несколько потомков, которыми являются узлы, соответствующие элементам меню нижнего уровня. При представлении меню в виде дерева листьями являются элементы множества $R = \{r\}$ – множество пунктов меню, активирующих команду, а узлами, имеющими потомков, – элементы множества $S = \{s\}$ – множество пунктов меню, содержащие вложенные подменю.

Каждый узел дерева – лист $r \in R$ дерева или узел $s \in S$, имеющий потомков, – относится к некоторому уровню иерархии навигационной структуры меню – $l^r$ или $l^s$ соответственно.

Пункт подменю, содержащий элемент $r \in R$, можно описать следующим образом:

$$s^r = (s \in S \mid s \prec r \in R,\ l^r - l^s = 1).$$

Эффективность работы пользователя с меню определяется глубиной уровня иерархии, на котором находится требуемый пользователю пункт меню. Поэтому целесообразно создавать ссылки на пункты меню, активированные $K$ раз, на более высоком уровне иерархии. При активации ранее созданной ссылки более $K$ раз создается новая ссылка на более высоком уровне. Таким образом, в модель меню добавляется множество $E$, содержащее ссылки на пункты меню. Если количество активаций пункта меню или ссылки на нее равно или превышает некоторое заданное количество $K$, то на один уровень выше пункта меню создается ссылка на этот пункт меню. Аналогичный принцип распространяется на ранее созданные ссылки при их многократной активации. Вновь созданная ссылка выделяется заметной меткой для того, чтобы пользователь обратил на нее внимание и начал ею пользоваться. Метка исчезает после того как пользователь ее активировал указанное количество раз (обычно один или два раза). В случае, если созданная ссылка не является востребованной (количество активаций менее порогового значения $K$ за указанное время $T$), то она удаляется. При расчете времени существования ссылок учитывается время работы пользователя в программе, а не календарное время.

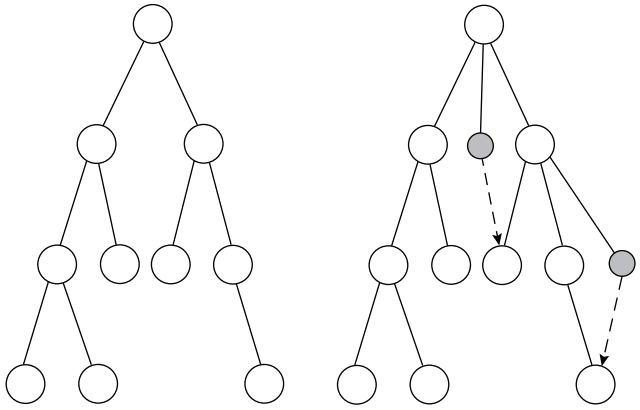

**Рис. 1.** Структура меню программного приложения в виде дерева до создания ссылок и в виде графа после создания ссылок.

Каждая ссылка $e \in E$ описывается парой $e = (r^e \in R, s^e \in S)$, где $r^e$ – пункт меню, на который указывает ссылка $e$; $s^e$ – подменю, включающее ссылку $e$.

Дерево меню с добавленными ссылками фактически превращается в граф, содержащий сокращенные пути к требуемым элементам меню, за счет обхода уровней меню (рис. 1).

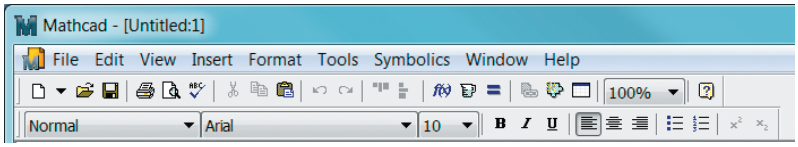


**Рис. 2.** Пример панелей инструментов в приложении PTC MathCAD.

## 3. МОДЕЛЬ ПОЛЬЗОВАТЕЛЯ ПРОГРАММНОГО ПРИЛОЖЕНИЯ

Для автоматической перенастройки меню под конкретного пользователя необходима модель пользователя, которая может строиться на основе наблюдений за действиями пользователя и фиксации того, сколько раз к элементам меню обращался пользователь [23, 24].

Пусть $U = \{u\}$ – множество пользователей. Тогда математическую модель отдельного пункта меню с учетом действий пользователя $u$ можно определить следующим образом:

– пункт меню $r$, активирующий команду, описывается пунктом меню, в котором он содержится, количеством активаций его пользователем $u$ и признаком видимости:

$$r = (s^r, v^r, h^r) \forall r \in R;$$

– ссылка на пункт меню описывается пунктом меню, являющимся объектом ссылки, пунктом меню, содержащим ссылку, количеством активаций ссылки пользователем $u$, временем ее создания:

$$e = (r^e, s^e, v^e, t^e) \forall e \in E.$$

Здесь $s^r = (s \in S \mid s \prec r, l^r - l^s = 1)$ – подменю, содержащее элемент $r$; $v^r$, $v^e$ – число активаций пользователем $u$ элемента меню $r$ и ссылки $e$ соответственно за указанное время работы с программным приложением; $h^r$ – признак видимости элемента $r$, принимающий одно из двух значений – “виден” (“visible”) или “скрыт” (“hidden”); $r^e = (r \in R \mid e \to r)$ – пункт меню, являющийся объектом ссылки $e$ (запись $e \to r$ означает, что $e$ является ссылкой на элемент $r$); $s^e$ – подменю, содержащее ссылку $e$; $t^e$ – время создания ссылки $e$.

Время обращения к пункту меню зависит от опыта пользователя, от уровня, на котором находится пункт меню, и количества элементов в подменю.

Степень соответствия навигационной структуры меню потребностям пользователя $u$ можно оценить средним временем доступа к элементам $r$, которое зависит от количества действий пользователя, необходимых для доступа к пункту меню, или от числа уровней навигационной структуры и места требуемого элемента в содержащем его подменю $s^r$.

Тогда задача перестройки меню сводится к такому изменению его навигационной структуры для пользователя $u$, при которой достигается минимум значения $\sum_{r \in R} v^r \cdot \tau^r$, где $\tau^r$ – среднее время активации пункта меню $r$. При этом накладываются ограничения на максимальное количество элементов, вынесенных из каждого подменю на один из уровней выше. Оно не должно превышать значения когнитивной константы $C$, определяющей условия эффективного восприятия пользователем элементов меню и комфортности его работы с программным приложением.

Решение данной задачи позволит получить оптимальную для конкретного пользователя навигационную структуру меню, обеспечивающую повышение комфортности и производительности работы пользователя.

## 4. АЛГОРИТМ АДАПТИВНОЙ ПОДСТРОЙКИ ПУНКТОВ МЕНЮ

Алгоритм предназначен для адаптивной подстройки меню к конкретному пользователю и должен обеспечить настройку меню с учетом действий пользователя, характеризуемых количеством активаций каждого пункта меню.

Целью предлагаемого алгоритма подстройки меню является сокращение суммарного времени обращения к элементам меню путем вынесения ссылок на часто используемые элементы меню на более высокий уровень иерархии и скрытия редко используемых пунктов меню.

Такой подход для ускорения обращения к командам реализуется с помощью панелей инструментов, обычно расположенных в окне программы под главным меню. Пример приведен на рис. 2.

Панели инструментов позволяют получить доступ к команде за один щелчок мыши, поскольку команды расположены непосредственно в окне формы. Но такое расположение “быстрых кно-

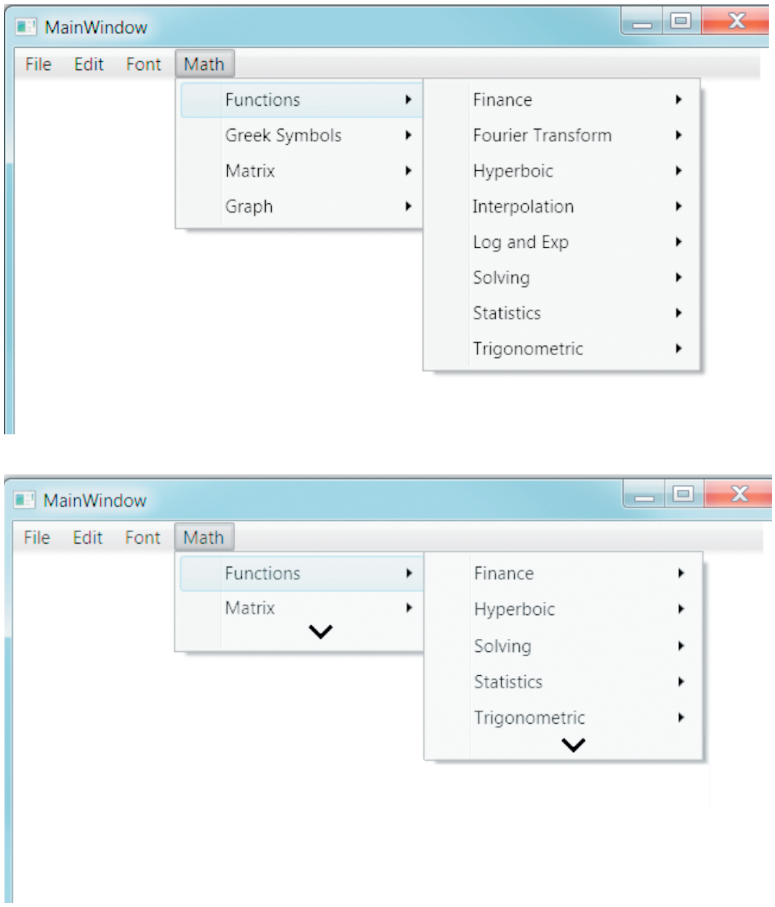


**Рис. 3.** Внешний вид меню программного приложения без скрытых пунктов меню и с усеченным вариантом меню.

пок” имеет свои недостатки: занимает место в окне, имеются кнопки, которые многим не нужны, требует ручной настройки для подгонки к пользователю, не всегда очевидно соответствие команды изображенной иконке. Даже автоматическое скрытие редко используемых команд в панели инструментов не всегда позволяет преодолеть выше перечисленные недостатки.

Предлагаемый алгоритм решает задачу сокращения времени активации команд путем перестройки меню под конкретного пользователя.

Количество уровней меню и количество пунктов в каждом подменю задается разработчиком изначально и в дальнейшем не изменяется.

Предлагаются два способа сокращения времени доступа:

1. Повышение уровня, на котором находится пункт меню, соответствующий часто используемой команде. С этой целью необходимо поместить ссылки на часто активируемые пункты меню на вышележащем уровне. Если ссылка также часто активируется, то следует создать ссылку на еще более высоком уровне и т.д.

2. Уменьшение количества пунктов меню: а) скрывать редко используемые ссылки на пункты меню (в Windows скрытые элементы можно увидеть при нажатии на кнопку раскрытия скрытой части подменю (рис. 3)); б) уничтожать редко используемые ссылки на пункты меню.

Предлагается пункты меню с большим количеством активаций дублировать на более высоком уровне с помощью разработанного алгоритма.

В общем случае адаптация меню для пользователя $u$ состоит в перераспределении элементов меню по уровням иерархии его навигационной структуры и в изменении видимости (“виден”/”скрыт”) элементов меню.

Входными данными алгоритма являются: модель меню программного приложения; модель пользователя программного приложения; когнитивная константа $C$; $K$ – количество активаций пункта меню, при котором необходимо создать ссылку; $V_{\min}$ – минимальное число активаций

пункта меню (пункт меню, для которого этот предел не достигнут, скрывается); $T$ – минимальное время существования ссылки, ранее которого она не может быть удалена.

Алгоритм подстройки меню программного приложения для текущего пользователя при активации пункта меню $r \in R$ включает следующие шаги:

1. Если пункт меню $r$ или подменю $s$, в котором находится пункт меню $r$, невидимы, то сделать их видимыми: $h^r = visible$, $h^s = visible$.

2. Если количество активаций пункта меню $r$ больше величины $K(v^r > K)$, то перейти к шагу 3, иначе перейти к шагу 5.

3. Определить, есть ли ссылка на пункт меню $r$ на вышестоящем уровне в подменю $\overline{s} = (s \in S \mid s \prec r, e \to r, l^r - l^s = 2)$. Если ссылка $\overline{e} = (e \in E \mid \overline{s} \prec e, l^e - l^{\overline{s}} = 1, e \to r)$ на пункт меню $r$ в подменю $\overline{s}$ имеется $(\overline{e} \neq \varnothing)$, то $t^{\overline{e}} = t_c$, $v^{\overline{e}} = 0$, где $t_c$ – текущее время, и закончить алгоритм.

4. Если $\overline{e} = \varnothing$, то создать ссылку $e$ на пункт меню $r$:

$$e = (r, (s \in S \mid s \prec r, l^r - l^s = 2), 0, t_c)$$

и занести $e$ в множество $E$: $E = E \cup e$.

5. Если количество ссылок в подменю $s$ больше заданного предельного значения $C$, т.е. $|e \in E \mid s \prec e, l^e - l^s = 1| > C$, то необходимо удалить ссылку $\overline{e}$ из подменю $s$, у которой время существования превышает указанный срок $T$ и за контролируемый период количество активаций минимально:

$$\overline{e} = (e \in E \mid s \prec e, l^e - l^s = 1, t_c - t^e > T, v^e = \min_e v^e)$$

$$E = E \backslash \overline{e}.$$

6. Если в множестве $E$ есть ссылки $E'$ с количеством активаций меньше $V_{\min}$, и со сроком существования более $T$, то необходимо их удалить:

$$E' = \{e \in E \mid t_c - t^e > T, v^e < V_{\min}\},$$

$$E = E \backslash E'.$$

7. Если есть пункты меню $r \in R$ с количеством активаций меньше $V_{\min}$, то необходимо их скрыть:

$$R' = \{r \in R \mid v^r < V_{\min}\},$$

$$h^r = hidden \ \forall r \in R'.$$

8. Если есть подменю, не содержащие других подменю, а включающие только элементы, активирующие команды, и количество активаций всех элементов указанных подменю меньше $V_{\min}$, то данные подменю необходимо скрыть:

$$S' = S\{s \in S \mid s \prec s', s' \in S\},$$

$$S'' = S' \backslash \{s \in S' \mid s \prec r \in R, v^r \geq V_{\min}\},$$

$$h^s = hidden \ \forall s \in S''.$$

Конец алгоритма.

Алгоритм адаптивной настройки меню программного приложения для текущего пользователя при активации ссылки $e$ на пункт меню $r$ отличается от предыдущего алгоритма шагами 1–3:

1. Если количество активаций ссылки $e$ на пункт меню $r$ больше величины $K(v^e > K)$, то перейти к шагу 2, иначе перейти к шагу 4.

2. Определить, есть ли ссылка на пункт меню $r$ на вышестоящем уровне в подменю $\overline{s} = (s \in S \mid e \to r, e \in E, r \in R, s \prec e, l^e - l^s = 2)$.

3. Если ссылка $\overline{e} = (e \in E \mid \overline{s} \prec e, l^e - l^{\overline{s}} = 1, e \to r)$ на пункт меню $r$ в подменю $\overline{s}$ имеется $(\overline{e} \neq \varnothing)$, то $t^{\overline{e}} = t_c$, $v^{\overline{e}} = 0$ и закончить алгоритм.

Описанные алгоритмы подстройки меню программного приложения для пользователя $u$ предлагается выполнять при каждой активации пункта меню или ссылки на пункт меню.

Рассмотренные алгоритмы можно адаптировать таким образом, чтобы они реализовывались не при каждой активации пункта меню, а при запуске программы или по таймеру.

Помимо выше описанных алгоритмов при каждом запуске программы осуществляется определение пунктов и подпунктов меню, которые необходимо скрыть. Скрываются пункты меню, которые не активировались в течение времени $T$ и пункты подменю, в которых отсутствуют подменю и каждый из находящихся в них пунктов меню не активировался:

$$h^r = hidden \ \forall r \in R \mid v^r = 0;$$

$$h^s = hidden \ \forall s \in S \mid s \in r, \quad r \in R, \quad \sum_{r \succ s} v^r = 0.$$

Непредсказуемая адаптация интерфейса может снизить удобство использования программного приложения, вызвать эмоциональную напряжен-

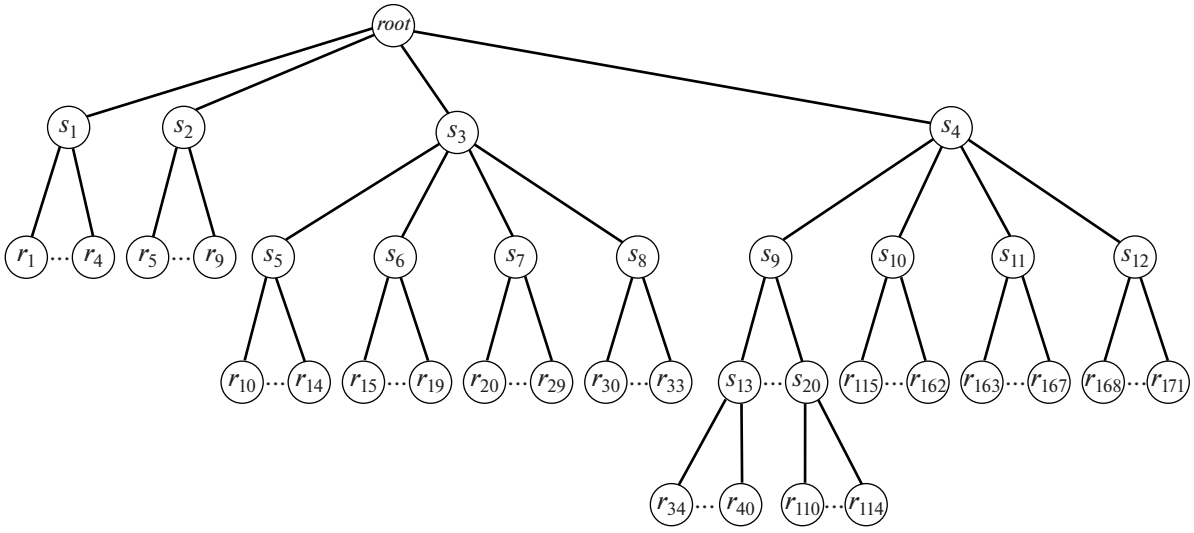


**Рис. 4.** Структура меню программного обеспечения до адаптивной подстройки.

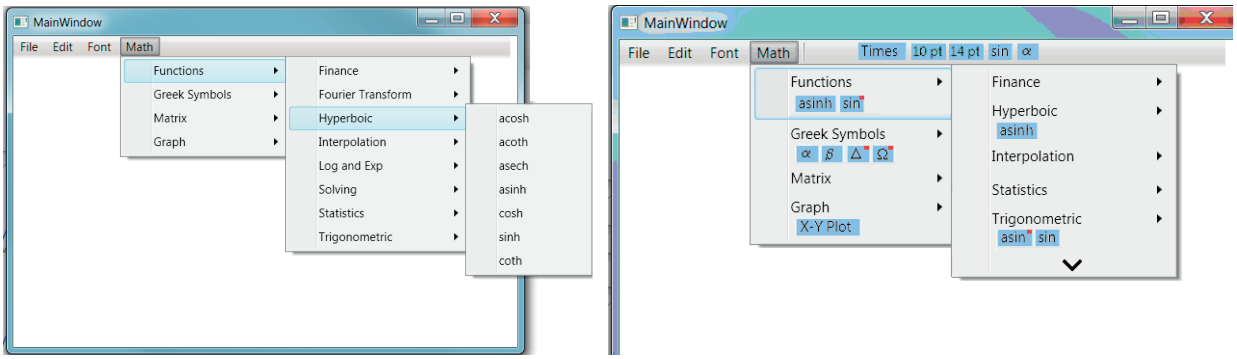


**Рис. 5.** Внешний вид меню программного обеспечения без ссылок и со ссылками.

ность пользователя и негативное восприятие неконтролируемых пользователем изменений интерфейса [5]. Для предотвращения этих проблем и повышения эргономичности меню программного приложения пользователю дается дополнительная возможность редактирования меню с учетом собственных предпочтений. Операции ручной подстройки пунктов меню активируются достаточно просто в режиме эксплуатации программного приложения без активации режимов настройки. Редактирование меню осуществляется устройством типа “мышь” при одновременном нажатии сочетания клавиш клавиатуры Ctrl/Shift/Alt. При этом пользователю предоставляется возможность:

1) выносить пункт меню или ссылку на пункт меню на уровень выше;

2) удалять ссылку на пункт меню;

3) менять положение пункта меню или ссылки на пункт меню, оставляя его на том же уровне;

4) делать пункт меню “видимым” или “невидимым”.

Изменения, внесенные пользователем, могут отменяться разработанным алгоритмом, но с согласия пользователя.

## 5. ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ АЛГОРИТМА

Тестирование разработанного алгоритма проводилось на тестовом программном обеспечении. Оно предназначено для набора форматированно-

**Таблица 1.** Результаты тестирования алгоритма

| Пункт меню | Уровень пункта меню | Количество пунктов по уровням от корня дерева без учета корня | Количество активаций пунктов меню | Среднее время активации пунктов меню при тестировании 1, с | Общее среднее время активации пунктов меню при тестировании 1, с | Уровни ссылок на пункты меню | Среднее время активации пунктов меню и ссылок на них при тестировании 2, с | Среднее время активации пунктов меню и ссылок на них при тестировании 3, с |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| $r_1$ | 3 | 8(4+4) | 3 | 1.54 | 4.32 | – | 1.52 | 1.54 |
| $r_2$ | 3 | 8(4+4) | 12 | 1.66 | 19.92 | 2 | 1.22 | 1.20 |
| $r_3$ | 3 | 8(4+4) | 67 | 1.62 | 108.54 | 2, 1 | 0.45 | 0.43 |
| $r_4$ | 3 | 8(4+4) | 1 | 1.63 | 1.53 | – | 1.61 | 1.60 |
| $r_9$ | 3 | 9(4+4) | 4 | 1.98 | 7.92 | – | 1.92 | 1.87 |
| $r_{11}$ | 4 | 13(4+4+5) | 14 | 2.81 | 51.8 | 3, 2 | 2.32 | 2.23 |
| $r_{12}$ | 4 | 13(4+4+5) | 1 | 2.82 | 3.8 | – | 2.81 | 2.71 |
| $r_{16}$ | 4 | 13(4+4+5) | 22 | 2.85 | 72.6 | 3, 2 | 2.21 | 2.10 |
| $r_{18}$ | 4 | 13(4+4+5) | 21 | 2.82 | 71.4 | 3, 2 | 2.24 | 2.22 |
| $r_{30}$ | 4 | 12(4+4+4) | 5 | 2.81 | 16.2 | 3 | 2.96 | 2.82 |
| $r_{31}$ | 4 | 12(4+4+4) | 2 | 2.88 | 6.46 | – | 2.87 | 2.72 |
| $r_{34}$ | 5 | 23(4+4+8+7) | 5 | 3.73 | 18.65 | – | 3.70 | 3.23 |
| $r_{35}$ | 5 | 23(4+4+8+7) | 6 | 3.80 | 22.8 | 4 | 3.68 | 3.57 |
| $r_{36}$ | 5 | 23(4+4+8+7) | 10 | 3.77 | 37.7 | 4, 3 | 3.40 | 3.28 |
| $r_{39}$ | 5 | 23(4+4+8+7) | 13 | 3.69 | 47.97 | 4, 3 | 3.31 | 3.12 |
| $r_{40}$ | 5 | 23(4+4+8+7) | 15 | 3.72 | 61.5 | 4, 3 | 3.25 | 3.10 |
| $r_{96}$ | 5 | 20(4+4+8+4) | 33 | 3.68 | 118.8 | 4, 3, 2, 1 | 0.96 | 0.95 |
| $r_{98}$ | 5 | 19(4+4+8+3) | 29 | 3.61 | 89.9 | 4, 3, 2, 1 | 1.25 | 1.22 |
| $r_{102}$ | 5 | 19(4+4+8+3) | 12 | 3.69 | 34.8 | 4, 3 | 2.51 | 2.33 |
| $r_{110}$ | 5 | 24(4+4+8+8) | 10 | 3.71 | 41 | 4, 3 | 2.42 | 2.31 |
| $r_{101}$ | 5 | 24(4+4+8+8) | 12 | 3.72 | 50.64 | 4, 3 | 2.43 | 3.33 |
| $r_{112}$ | 5 | 24(4+4+8+8) | 11 | 3.87 | 45.87 | 4, 3 | 2.39 | 2.34 |
| $r_{115}$ | 4 | 56(4+4+48) | 30 | 3.22 | 96.6 | 3, 2, 1 | 1.29 | 1.23 |
| $r_{116}$ | 4 | 56(4+4+48) | 30 | 3.41 | 102.3 | 3, 2, 1 | 1.31 | 1.28 |
| $r_{118}$ | 4 | 56(4+4+48) | 3 | 3.43 | 10.29 | – | 3.45 | 3.44 |
| $r_{138}$ | 4 | 56(4+4+48) | 12 | 3.24 | 38.88 | 3, 2 | 2.41 | 2.21 |
| $r_{142}$ | 4 | 56(4+4+48) | 4 | 3.25 | 13 | – | 3.23 | 3.23 |
| $r_{162}$ | 4 | 56(4+4+48) | 22 | 3.36 | 73.92 | 3, 2, 1 | 1.63 | 1.57 |
| $r_{168}$ | 4 | 12(4+4+4) | 7 | 3.2 | 22.4 | 3 | 3.01 | 3.00 |

го текста и математических формул, выбираемых из пунктов главного меню. Структура меню представлена на рис. 4.

Меню программного обеспечения содержит 171 элемент, каждый из которых соответствует пункту меню, активирующему команду. Иерархическая структура меню представляется деревом с глубиной, равной 5. Внешний вид меню программы до и после создания ссылок представлен на рис. 5. Выделение цветом ссылок позволяет визуально отделить их от пунктов меню и тем самым время доступа к пунктам меню и ссылкам практически не увеличивается из-за суммарного увеличения количества пунктов меню и ссылок. Таким образом, в каждом подменю $s \in S$ достаточно обеспечить $|r \in R \mid r \succ s \in S, l^r - l^s = 1| \le C$ и $|e \in E \mid e \succ s \in S, l^e - l^s = 1| \le C$. Кроме того, недавно созданные ссылки на пункты меню, которые активировались не более двух раз, дополнительно выделяются меткой (в правом верхнем углу ссылок на рис. 5), а пункты меню, которые не активировались дольше времени $T$, и подпункты меню, в которых каждый пункт меню не активировался ни разу, скрываются.

Для оценки эффективности предлагаемого алгоритма пользователям в количестве 36 человек было выдано задание на ввод в программу текста за минимально возможное время. По мере набора текста необходимо было активировать пункты меню, указанные в задании. В ходе тестирования пользователи должны были выполнить 416 активаций 29 пунктов меню, описанных в столбцах 1–4 табл. 1. В табл. 1 приведены только пункты меню, активированные хотя бы один раз. Ошибочно активированные пользователями пункты меню в таблицу не включены.

Для оценки эффективности предложенных алгоритмов было проведено три тестирования с интервалом в одну неделю с привлечением к тестированию одних и тех же пользователей.

Тестирование 1 проводилось без создания ссылок на пункты меню. Всего на выполнение задания было потрачено в среднем 2642 с. Общее время активации всех указанных в задании пунктов меню составило в среднем на одного пользователя 1261.55 с. Результаты тестирования 1 приведены в столбцах 5, 6 табл. 1.

Тестирование 2 было проведено с активацией алгоритма для создания ссылок на пункты меню. Общее время тестирования было уменьшено до 2127 с в среднем на одного пользователя, т.е. на 19.5%. В результате время активации пунктов меню и ссылок на них составило 741.31 с, что сэкономило 4123% времени активации пунктов меню и ссылок на них. Уровни ссылок на пункты меню и результаты тестирования 2 приведены в столбцах 7, 8 табл. 1.

Тестирование 3 проводилось с активацией алгоритма с созданием ссылок на пункты меню и предварительным скрытием некоторого количества заведомо ненужных пунктов меню. В результате время активации пунктов меню уменьшилось до 724.72 с в среднем на одного пользователя, т.е. на 42.55%. Результаты тестирования приведены в столбце 9 табл. 1.

## 6. ЗАКЛЮЧЕНИЕ

В статье предложен подход к адаптации меню программного приложения к потребностям конкретного пользователя посредством добавления ссылок на часто используемые пункты меню на более высоких уровнях его навигационной структуры. Результаты свидетельствуют о получении практического эффекта в виде уменьшения времени на активацию пунктов меню. Разработанный алгоритм и его программная реализация позволяют создать программное приложение с адаптивным меню, повышающим эффективность работы пользователя, заключающуюся в экономии времени обращения к пунктам меню.

Предлагаемый в данной статье подход к построению адаптивного меню характеризуется следующими особенностями.

1. На часто используемые элементы меню создаются ссылки, расположенные на уровень выше, вместо переноса этих элементов в начало списка на том же уровне в отличие от [8].

2. Ссылки выделяются визуально, что позволяет рассматривать их в качестве “параллельного” набора команд и независимо применять к этому набору накладываемое когнитивной константой ограничение.

3. Наряду с автоматической адаптацией предложены способы ручной настройки меню пользователем без необходимости входа в специальный режим настройки. Данное решение представляет комбинацию свойств адаптивных и адаптируемых меню.

Дальнейшее исследование предполагает изучение возможности повышения эффективности работы пользователя программного приложения за счет повышения адаптационных свойств других элементов пользовательского интерфейса.

## СПИСОК ЛИТЕРАТУРЫ


1. *López-Jaquero V., Montero F., Molina J.P., Fernández-Caballero A., González P.* Model-Based Design of Adaptive User Interfaces through Connectors // Interactive Systems. Design, Specification, and Verification (DSV-IS 2003). Lecture Notes in Computer Science, Springer, Berlin, Heidelberg. 2003. V. 2844. P. 245–257.
2. *Paymans T.F., Lindenberg J., Neerincx M.* Usability trade-offs for adaptive user interfaces: Ease of use and learnability // Proceedings of the 9th International Conference on Intelligent User Interfaces (IUI '04). 2004. P. 301–303.
3. *Peissner M., Schuller A., Spath D.* A Design Patterns Approach to Adaptive User Interfaces for Users with Special Needs // Human-Computer Interaction. Design and Development Approaches (HCI 2011). Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2011. V. 6761. P. 268–277.
4. *Zosimov V.V., Khrystodorov O.V. & Bulgakova O.S.* Dynamically Changing User Interfaces: Software Solutions Based on Automatically Collected User Information // Programming and Computer Software. 2018. V. 44. P. 492–498.
5. *Takacs B., Simon L.* Sensing User Needs: Recognition Technologies and User Models for Adaptive User Interfaces // Human-Computer Interaction. Design and Development Approaches (HCI 2011). Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 2011. V. 6761. P. 498–506.
6. *Nazemi K., Stab C., Fellner D.W.* Interaction Analysis for Adaptive User Interfaces // Advanced Intelligent Computing Theories and Applications (ICIC 2010). 2010. V. 6215. P. 362–371.
7. *Alvarez-Cortes V., Zarate V.H., Uresti J.A.R., Zayas B.E.* Current Challenges and Applications for Adaptive User Interfaces // Human-Computer Interaction. 2009. P. 13–30.
8. *Sears A., Shneiderman B.* Split menus: Effectively using selection frequency to organize menus // ACM Transactions on Computer-Human Interaction. 1994. V. 1. № 5. P. 27–51.
9. *Soui M., Chouchane M., Mkaouer M.W., Kessentini M., Ghedira K.* Assessing the quality of mobile graphical user interfaces using multi-objective optimization // Soft Computing. 2019. P. 1–30.
10. *Sluis-Thiescheffer R.J.W., Bekker M.M., Eggen J.H., Vermeeren A.P.O.S., De Ridder H.* Development and application of a framework for comparing early design methods for young children // Interacting with Computers. 2011. V. 23. № 1. P. 70–84.
11. *Akiki P.A., Bandara A.K., Yu Y.* Adaptive Model-Driven User Interface Development Systems // ACM Computing Surveys (CSUR). 2014. V. 47. № 1. Article no 9. 33 p.
12. *Findlater L., McGrenere J.* A Comparison of Static, Adaptive, and Adaptable Menus // ACM Conference on Human Factors in Computing (CHI 2004). 2004. V. 6. № 1. P. 89–96.
13. *Park J., Han S.H.* Integration of Adaptable and Adaptive Approaches for Interface Personalization through Collaborative Menu // International Journal of Human-Computer Interaction. 2012. V. 28. № 9. P. 613–626.
14. *Elio R., Haddadi A.* Dialog management for an adaptive database assistant (Technical Report 98–3). Daimler-Benz Research and Technology Center, Palo Alto, CA. 1998.
15. *Zhang L., Qu Q.-X., Chao W.-Y., Duffy V.G.* Investigating the combination of adaptive UIs and adaptable UIs for improving usability and user performance of complex UIs // International Journal of Human-Computer Interaction. 2020. V. 36. № 1. P. 82–94.
16. *Hervás R., Bravo J.* Towards the ubiquitous visualization: Adaptive user-interfaces based on the Semantic Web // Interacting with Computers. 2011. V. 23. № 1. P. 40–56.
17. *Park K., Lee S.W.* Model-Based Approach for Engineering Adaptive User Interface Requirements // Requirements Engineering in the Big Data Era. Communications in Computer and Information Science. Springer, Berlin, Heidelberg. 2015. V. 558. P. 18–32.
18. *Schlimmer J.C., Hermens L.A.* Software Agents: Completing Patterns and Constructing User Interfaces // Journal of Artificial Intelligence Research. 1993. V. 1. P. 61–89.
19. *Mitchell J., Shneiderman B.* Dynamic versus static menus: an exploratory comparison // SIGCHI Bull. 1989. V. 20. № 4. P. 33–37.
20. *McGrenere J., Baecker R., Booth K.* An evaluation of a multiple interface design solution for bloated software // Processing of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02). Association for Computing Machinery, New York, NY, USA. 2002. V. 4. № 1. P. 164–170.
21. *Shneiderman B.* Direct manipulation for comprehensible, predictable and controllable user interfaces // Intelligent User Interfaces (IUI '97). 1997. P. 33–39.
22. *Matsui S., Yamada S.* Optimizing hierarchical menus by genetic algorithm and simulated annealing // Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08). 2008. P. 1587–1594.

23. *Abascal J., Aizpurua A., Cearreta I., Gamecho B., Garay N., Miñón R.* Some Issues Regarding the Design of Adaptive Interface Generation Systems // Proceedings of the 6th International Conference on Universal Access in Human-Computer Interaction. Design for All and inclusion (UAHCI 2011). Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg. 2011. V. 6765. P. 307–316.

24. *Langley P.* Machine learning for adaptive user interfaces // KI-97: Advances in Artificial Intelligence. Lecture Notes in Computer Science. Springer, Berlin, Heidelberg. 1997. V. 1303. P. 53–62.