DOI 10.37882/2223-2966.2024.06.13

# ЭРГОНОМИЧЕСКИЙ АНАЛИЗ РЕКОМЕНДАТЕЛЬНОЙ СИСТЕМЫ

## ERGONOMIC ANALYSIS OF A RECOMMENDATION SYSTEM

***B. Goryachkin***
***R. Nischuk***

*Summary*. The complexity of choosing books from a vast amount of information poses a serious problem for users. The need for an ergonomic analysis of recommendation systems is due to the unsatisfactory user experience in finding the most suitable reading options. Ineffective interaction with the system leads to time loss, insufficient information, and dissatisfaction with the service usage.

The process of selecting books using recommendation systems involves analyzing user data and comparing it with a large amount of book information to choose the most suitable options. This significantly saves user time, optimizes the selection process, and increases satisfaction with the service usage.

Through the example of a recommendation system for book selection, one can see how a balance between technical complexity and users can be achieved through thoughtful design, intuitive controls, and personalized recommendations.

*Keywords*: recommendation system, ergonomics, Django, Big Data.

***Горячкин Борис Сергеевич***
*кандидат технических наук, доцент,*
*Московский государственный технический университет им. Н.Э. Баумана*
*bsgor@mail.ru*
***Нищук Роман Сергеевич***
*Московский государственный технический университет им. Н.Э. Баумана*
*gidraaa@inbox.ru*

*Аннотация*. Сложность выбора книг из огромного объема информации представляет серьезную проблему для пользователей. Необходимость проведения эргономического анализа рекомендательных систем обусловлена неудовлетворительным опытом пользователей в поиске наиболее подходящих вариантов чтения. Отсутствие эффективного взаимодействия с системой приводит к потере времени, недостаточной информированности и неудовлетворенности от использования сервиса.

Процесс подбора книг с использованием рекомендательных систем включает анализ данных пользователя и их сравнение с большим объемом информации о книгах для выбора наиболее подходящих вариантов. Это значительно экономит время пользователя, оптимизирует процесс выбора и повышает удовлетворенность от использования сервиса.

На примере рекомендательной системы для подбора книг можно увидеть, как баланс между технической сложностью и пользователями может быть достигнут через продуманный дизайн, интуитивное управление и персонализацию предложений.

*Ключевые слова*: рекомендательная система, эргономика, Django, Big Data.

## Введение

Рекомендательные системы используются практически везде. Наиболее популярными областями для их применения является подбор музыки, товаров для дома, продуктов или фильмов по предпочтениям конкретных пользователей. Тем не менее рекомендательные системы могут использоваться практически в любой области. Система может рекомендовать приложения, новости для прочтения, видео и даже лекарства. От качества этих систем напрямую зависит популярность сервиса: как долго пользователь будет использовать сервис или как много купит в магазине.

Основной задачей, поставленной в данной работе, является изучение и эргономический анализ программного модуля для подбора книг. Рекомендательная система, лежащая в основе модуля, может давать пользователю советы по выбору литературы, которые будут зависеть от активности пользователя и постоянно меняться. [1]

Эргономический анализ данных в контексте рекомендательных систем представляет собой интересную и актуальную область исследования, требующую внимания с точки зрения практической применимости и развития.

В данной работе будет проведен эргономический анализ автоматизированной рекомендательной системы по подбору книг в формате стандартного Django-приложения, что позволит оценить ее удобство использования и качество предоставляемых рекомендаций. Анализ проведенных исследований позволит определить оптимальные пути развития и улучшения функциональности данной системы, удовлетворяя потребности пользователей в максимальной степени.

## Эргономика в контексте рекомендательных систем

Методической базой эргономики служит системный подход. На его основе в эргономических исследованиях используются методы различных наук и техники, на стыке которых решаются проблемы изучения системы «человек — предмет — среда». Метод функционального анализа может быть использован при проектировании относительно несложных объектов. Он вскрывает зави-

симость формы, структуры, организации и конструкции объекта от той функции, которую он выполняет. Используется и при проектировании средовых комплексов. [8]

Основные аспекты эргономики для рекомендательных систем включают в себя:

- *Пользовательский интерфейс:* Создание удобного и интуитивно понятного интерфейса, который позволяет пользователям легко взаимодействовать с системой, предоставлять обратную связь и управлять рекомендациями.
- *Персонализация:* учитывать индивидуальные предпочтения, интересы и контекст каждого пользователя для предоставления более точных и релевантных рекомендаций.
- *Прозрачность и объяснимость:* объяснять пользователю принципы работы системы рекомендаций, чтобы он понимал, почему та или иная рекомендация была сделана, и мог доверять системе.
- *Контроль и настройка:* предоставлять пользователям возможность контролировать процесс рекомендаций, например, путем настройки параметров алгоритмов или фильтрации рекомендаций.
- *Управление информационной нагрузкой:* избегать перегрузки пользователей информацией и рекомендациями, предоставляя только наиболее релевантные и полезные рекомендации.

Цель эргономики в контексте рекомендательных систем состоит в создании приятного и продуктивного опыта использования для пользователей, что в конечном итоге способствует повышению удовлетворенности пользователей и эффективности системы. [2]

Для дальнейшего понимания опишем процесс подбора книг с помощью формулы для человека и машины:

**Человек:**

- Просмотр каталога: Время, затраченное на просмотр доступных книг в каталоге.
- Чтение описаний: Время, потраченное на чтение описаний книг для определения их соответствия интересам.
- Принятие решения: Время, потраченное на принятие решения о выборе книги на основе прочитанных описаний и собственных предпочтений.

**Машина:**

- Анализ данных: Время, необходимое для анализа данных о предпочтениях пользователя, истории просмотров, рейтингах книг и т. д.
- Сравнение с базой данных: Время, затраченное на сравнение профиля пользователя с базой данных книг для определения наиболее подходящих рекомендаций.
- Формирование рекомендаций: Время, необходимое для сортировки и фильтрации книг с целью предоставления наиболее релевантных рекомендаций пользователю.

Формула для оценки времени может выглядеть следующим образом:

$$Время\ подбора = Время\ анализа + Время\ принятия\ решения$$

где:

- Время анализа для человека включает в себя просмотр каталога и чтение описаний книг, а для машины — анализ данных и сравнение с базой данных.
- Время принятия решения для человека — время, потраченное на принятие окончательного решения, а для машины — время формирования рекомендаций.

В анализ информации о книге может входить множество аспектов, включая:

- *Описание книги:* Это текстовое описание содержания книги, которое может содержать краткое изложение сюжета, основные темы, персонажей и другие ключевые моменты.
- *Обложка книги:* Визуальный элемент, который также может оказать влияние на решение пользователя. Обложка может привлечь внимание своим дизайном, цветами, изображениями и т. д.
- *Рейтинги и отзывы:* Рейтинги книги на различных платформах, таких как интернет-магазины или рецензии на сайтах и блогах, могут дать представление о том, насколько книга популярна и понравилась другим читателям. Отзывы также могут содержать полезную информацию о содержании, стиле и качестве книги.
- *Жанр и тематика:* Пользователь может обращать внимание на жанр или тему книги, чтобы определить, соответствует ли она его интересам или текущим настроениям.
- *Цена и доступность:* Цена книги и ее доступность в выбранном формате (бумажная книга, электронная книга и т. д.) также могут влиять на решение пользователя.
- *Автор:* Информация о авторе, его репутация и предыдущие работы могут быть также важными факторами при выборе книги.

Время, необходимое для обработки информации человеческим сознанием, может сильно варьироваться в зависимости от объема информации, сложности задачи, индивидуальных способностей и опыта человека. Рассмотрим несколько примеров с различными объемами информации: [4]

- *Небольшой объем информации (несколько книг):*

Если человеку предстоит выбрать книгу из небольшого набора, состоящего из, скажем, 5-10 книг, то время на обработку информации может быть относительно небольшим. В этом случае процесс выбора книги может занять от нескольких минут до 10-15 минут в зависимости от внимательности и скорости принятия решений человека.

- *Средний объем информации (десятки книг):*

Если доступен более крупный набор книг, например, от 20 до 50 книг, то время на обработку информации увеличится. Человеку потребуется больше времени на просмотр обложек, чтение описаний и анализ отзывов. В этом случае процесс выбора книги может занять от 20 минут до часа или более, в зависимости от того, насколько детально человек изучает каждую книгу.

- *Большой объем информации (сотни книг):*

Если доступен очень большой каталог книг, содержащий сотни или даже тысячи позиций, время на обработку информации значительно возрастет. В этом случае человек может потребовать нескольких часов или даже дней на анализ каталога, чтение описаний и принятие решения. Некоторые люди могут использовать различные стратегии, такие как фильтрация по жанрам или авторам, чтобы сократить время на выбор.

Таким образом, суммарное время подбора книги для среднестатистического пользователя при малом объёме информации может быть приблизительно:

Для того, чтобы приблизительно узнать суммарное время подбора книги для среднестатистического пользователя при малом объёме информации проведём экспериментальный опрос 10 добровольцев на малом наборе данных из 10 классических книг русской литературы:

*«Преступление и наказание» — Федор Достоевский*
*«Война и мир» — Лев Толстой*
*«Мастер и Маргарита» — Михаил Булгаков*
*«Евгений Онегин» — Александр Пушкин*
*«Анна Каренина» — Лев Толстой*
*«Братья Карамазовы» — Федор Достоевский*
*«Доктор Живаго» — Борис Пастернак*
*«Отцы и дети» — Иван Тургенев*
*«А зори здесь тихие...» — Борис Васильев*
*«Тихий Дон» — Михаил Шолохов*

Ход проведения эксперимента выглядит следующим образом: Проведение эксперимента:

1. Каждый доброволец будет иметь доступ к списку книг.
2. Дадим каждому добровольцу случайную книгу из списка и попросим их проанализировать информацию о книге.
3. Когда они закончат анализ, попросим их принять решение: стоит ли рекомендовать эту книгу кому-то или нет.
4. Измерим время, затраченное каждым добровольцем на анализ и принятие решения.

Измерения времени [в секундах]:

$t_{ан}$ — время анализа набора данных из 10 книг
$t_{реш}$ — время принятия решения о рекомендации

Результаты опроса приведены в табл. 1.

Таблица 1.

Результаты экспериментального опроса

| Добр. | №1 | №2 | №3 | №4 | №5 | №6 | №7 | №8 | №9 | №10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $t_{ан}$ | 330 | 612 | 1128 | 642 | 546 | 726 | 540 | 864 | 468 | 792 |
| $t_{реш}$ | 108 | 234 | 156 | 138 | 54 | 45 | 192 | 162 | 114 | 72 |

Среднее значение $t_{ан} = 7048 / 10 = 704.8 \approx 11{,}75$ минут

Среднее значение $t_{реш} = 1165 / 10 = 116.5 \approx 1{,}94$ минуты

В данной работе мы постараемся значительно уменьшить экспериментально измеренное приблизительное время на подбор подходящей книги, при том, что мы используем для анализа малый набор книг из 10 позиций, а рекомендательная система будет анализировать датасет из 6810 книг.

## Аналитическая оценка времени выполнения блоков кода рекомендательной системы

Для начала следует разобраться в том, как именно работает рекомендательная система. Приведём схему работы модулей системы по отдельности и распишем что происходит в каждом из блоков кода, подлежащих анализу рис. 1:

При запуске веб-приложения Django происходит следующий порядок обращения к файлам:

- **`apps.py`:** Файл `apps.py` содержит классы приложений Django. Django автоматически обнаруживает приложения, указанные в этом файле, и регистрирует их в проекте.
- **`models.py`:** Файл `models.py` содержит описания моделей данных приложения. Модели определяют структуру данных, которые будут использоваться в приложении, и обеспечивают взаимодействие с базой данных.
- **`admin.py`:** Файл `admin.py` содержит настройки административного интерфейса Django. Здесь


Рис. 1. Последовательность блоков кода Django-приложения

определяются модели, которые должны быть доступны в админ-панели, а также их настройки.

- **`views.py`:** Файл `views.py` содержит функции или классы представлений Django. Представления определяют логику обработки запросов и возвращают ответы на эти запросы. Они связываются с URL-шаблонами для определения, какой код должен быть выполнен при обращении к определенному URL.
- **`forms.py`:** Файл `forms.py` содержит определения форм Django. Формы используются для ввода данных пользователем и их валидации перед сохранением.
- **`urls.py`:** Файл `urls.py` содержит определения URL-шаблонов Django. URL-шаблоны связывают определенные URL с соответствующими представлениями, определенными в файле `views.py`.
- **`tests.py`:** Файл `tests.py` содержит модули тестирования Django. Здесь определяются тесты, которые проверяют правильность работы приложения. [3]

Для оценки времени выполнения каждого из приведённых блоков кода можно использовать адаптированную формулу, учитывающую размер набора данных и другие факторы:

$$T(n) = e^{a*n} * b * c * d$$

Где:

- $T(n)$ — время выполнения рекомендательной системы в зависимости от размера набора данных или других факторов.
- $n$ — размер набора данных, например, количество пользователей и элементов для рекомендации.
- $a$ — коэффициент, который определяет экспоненциальную зависимость времени выполнения системы от размера набора данных.
- $e$ — база натуральных логарифмов (постоянная Эйлера).
- $b$ — множитель, представляющий базовое время выполнения системы для небольших наборов данных.
- $c$ — дополнительный множитель, учитывающий специфические аспекты алгоритма рекомендации, такие как сложность вычислений или используемые модели.
- $d$ — дополнительный множитель, который показывает мощность конфигурации системы, на которой запускается обработка набора данных.

Следует объяснить почему именно выбрана экспоненциальная зависимость. Сделано это по следующим причинам:

- *Экспоненциальный рост данных*: Когда размер набора данных ($n$) увеличивается, время выполнения рекомендательной системы может расти экспоненциально. Это связано с увеличением вычислительной сложности алгоритмов и операций обработки данных.
- *Увеличение вычислительной нагрузки*: С увеличением объема данных система должна обрабатывать больше информации, выполнять больше вычислений и операций. Это может привести к экспоненциальному росту времени выполнения.
- *Сложность алгоритмов*: Некоторые алгоритмы, используемые в рекомендательных системах, могут иметь высокую вычислительную сложность, особенно при увеличении размера данных. Это может вызывать экспоненциальный рост времени выполнения с увеличением размера набора данных.

- *Увеличение количества операций*: При обработке больших объемов данных требуется выполнение большего количества операций, таких как сравнения, вычисления сходства между элементами, что может привести к экспоненциальному увеличению времени выполнения. [5]

Перейдём к расчётам относительного времени выполнения каждого блока кода в стандартном Django-приложении на основе представленной формулы:

$$T(n) = e^{a*n} * b * c * d$$

$n$ = 6810 (количество книг в наборе данных)

1. apps.py: Время выполнения кода в файле apps.py, представляющего конфигурацию Django-приложения, может быть относительно небольшим, так как это обычно простой файл с настройками.

Пусть ***(a = 0.0001), (b = 0.1), (c = 2), (d = 1).***

2. models.py: Время выполнения кода в файле models.py, где определяются модели данных приложения, может зависеть от сложности моделей и количества полей.

Пусть ***(a = 0.0005), (b = 0.2), (c = 3), (d = 1).***

3. admin.py: Файл admin.py, содержащий настройки административного интерфейса Django, обычно не требует значительного времени выполнения.

Пусть ***(a = 0.00005), (b = 0.05), (c = 1.5), (d = 1)***.

4. views.py: Время выполнения кода в файле views.py, где определены представления (views) для обработки запросов, может зависеть от сложности логики обработки запросов и количества представлений.

Пусть ***(a = 0.0008), (b = 0.3), (c = 4), (d = 1).***

5. forms.py: Файл forms.py, где определяются формы для взаимодействия с пользователем, обычно не требует значительного времени выполнения.

Пусть ***(a = 0.00006), (b = 0.06), (c = 1.2), (d = 1).***

6. urls.py: Время выполнения кода в файле urls.py, где определяются маршруты URL для приложения, может быть относительно небольшим, особенно для небольших проектов.

Пусть ***(a = 0.00003), (b = 0.03), (c = 1.1), (d = 1).***

7. tests.py: Время выполнения кода в файле tests.py, где определяются тесты для проверки функциональности приложения, может быть значительным в зависимости от количества и сложности тестов. Так как здесь используется параллельная обработка значение коэффициента $d = 2$.

Пусть ***(a = 0.001), (b = 0.4), (c = 5), (d = 2). [6]***

Проведём расчёты:

$$T_{apps.py} = e^{0.0001*6810} * 0,1 * 2 * 1 \approx 1,976 * 0,1 * 2 \approx 0,3952$$

$$T_{models.py} = e^{0.0005*6810} * 0,2 * 3 * 1 \approx$$
$$\approx 30,114 * 0,2 * 3 \approx 18,0684$$

$$T_{admin.py} = e^{0.00005*6810} * 0,05 * 1,5 * 1 \approx$$
$$\approx 1,405 * 0,05 * 1,5 \approx 0,1054$$

$$T_{views.py} = e^{0.0008*6810} * 0,3 * 4 * 1 \approx$$
$$\approx 231,735 * 0,3 * 4 \approx 277,682$$

$$T_{forms.py} = e^{0.00006*6810} * 0,06 * 1,2 * 1 \approx$$
$$\approx 1,504 * 0,06 * 1,2 \approx 0,1083$$

$$T_{urls.py} = e^{0.00003*6810} * 0,03 * 1,1 * 1 \approx$$
$$\approx 1,226 * 0,03 * 1,1 \approx 0,0402$$

$$T_{tests.py} = e^{0.001*6810} * 0,4 * 5 * 2 \approx$$
$$\approx 900,139 * 0,4 * 5 * 2 \approx 3600,556$$

Все полученные результаты получены в относительных величинах времени, для сравнения времени выполнения [7]

Занесём полученные результаты в табл. 2 для дальнейшего удобства:

Таблица 2.

Результаты теоретических вычислений времени выполнения блоков кода

| | apps.py | models.py | admin.py | views.py | forms.py | urls.py | tests.py |
|---|---|---|---|---|---|---|---|
| T(n) | 0,3952 | 18,0684 | 0,1054 | 277,682 | 0,1083 | 0,0402 | 3600,556 |

Далее проведём экспериментальную проверку полученных значений с помощью измерений времени выполнения каждого блока кода напрямую:

```
Измерение времени выполнения блоков кода:
import time

# Начало измерения времени
start_time = time.time()

# блок кода, время выполнения которого нужно измерить
# ...

# Конец измерения времени
end_time = time.time()

# Вычисление общего времени выполнения
execution_time = end_time — start_time

print(f»Время выполнения: {execution_time} секунд»)
```

Полученные значения [в секундах]: apps.py: 0.00289946516192

models.py: 1.00702908464
admin.py: 0.00064548183
views.py: 1.207333592
forms.py: 0.00057923568
urls.py: 0.0003320199
tests.py: 48.0042411188

Вычисленные значения приведены в табл. 3.

Таблица 3.

Вычисленные значения времени выполнения блоков кода

| Значения | apps.py | models.py | admin.py | views.py | forms.py | urls.py | tests.py |
|---|---|---|---|---|---|---|---|
| Анализ | 0,3952 | 18,0684 | 0,1054 | 277,682 | 0,1083 | 0,0402 | 3600,556 |
| Эксперимент | 0.00289 | 1.0070 | 0.00064 | 1.2073 | 0.00057 | 0.00033 | 48.00424 |

Для наглядности приведём доли времени выполнения блоков кода в табл. 4.

Таблица 4.

Вычисленные доли времени выполнения блоков кода

| Доли | apps.py | models.py | admin.py | views.py | forms.py | urls.py | tests.py |
|---|---|---|---|---|---|---|---|
| Анализ | 0.000109 | 0.00501 | 0.00002 | 0.07709 | 0.00003 | 0.00001 | 1.000 |
| Эксперимент | 0.00006 | 0.02097 | 0.00001 | 0.07297 | 0.00001 | 0.000006 | 1.000 |

Эксперимент подтверждает аналитическое предположение и приведённую ранее формулу. Безусловно, конкретные доли и значения отличаются от проведённого анализа, но общая картина остаётся всё той же. Наибольшее время занимает выполнение tests.py и views.py.

Таким образом, получаем наглядную картину того, что наибольшее время выполнения приходится на блок тестирования и визуализации. Блок тестирования является необходимым только на этапе разработки, поэтому напрямую пользователя не затрагивает. Это означает что при грамотной наладке приложения данный блок кода не будет влиять на время выполнения пользовательского запроса.

Вопрос же визуальной составляющей целиком и полностью ложится на мощности компьютера пользователя, сервер разработчика и требования дизайнерского решения к конфигурации компьютерной системы.

## Эргономический анализ пользовательского интерфейса

Эргономическое обеспечение проектирования означает установление оптимальных требований и создание подходящих свойств системы «человек — предмет» на всех этапах её разработки и использования. Это включает в себя не только рассмотрение взаимодействия между человеком и предметом, но также влияние окружающей среды на этот процесс. Главная цель эргономического проектирования заключается в том, чтобы сделать изделия максимально эффективными при минимальном использовании ресурсов человека. Это включает в себя оптимизацию условий труда, уменьшение рисков для здоровья и безопасности человека, а также повышение удовлетворенности пользователей. [8]

В рамках этой концепции эргономическое проектирование также стремится учитывать различные особенности пользователей, такие как пол, возраст, физические и когнитивные способности. Например, изготовление предметов с учётом различий в организме мужчин и женщин, а также удобство использования для людей разных возрастных групп.

Основной задачей эргономического анализа данных является выявление факторов, влияющих на удобство, безопасность и эффективность использования продуктов, систем и услуг. Это может включать в себя анализ психофизиологических реакций пользователей, оценку эффективности дизайна, а также оценку уровня комфорта. [9]

В данном разделе попытаемся максимально оптимизировать время работы с рекомендательной системой человеком-оператором, в нашем случае обозначим его как пользователя.

В ходе поиска нужной книги для её дальнейшего анализа (прочтения описания, просмотра обложки и т.д.) человек выполняет ряд типовых действий:

1. Нажатие клавиши компьютерной мышки;
2. Перемещение курсора компьютерной мышки;
3. Набор текста с помощью клавиатуры.

Каждое из перечисленных действий состоит из времени его выполнения и времени реакции человека и системного времени.

Для возможности дальнейших подсчётов зададим ограничения среды:

- Время нажатия клавиши мышки составляет 180 мс
- Средняя скорость печати составляет 330 знаков в минуту
- *DPI* мышки = 1600 (9)
- Скорость перемещения мыши по столу = 2 см/с
- Задержка перед выполнением задания (a) = 0;
- Коэффициент скорости принятия решения (b) = 0,155;
- Время открытия всплывающего окна — 0,5 с.
- Начальное положение курсора находится в центре экрана.

На основе измеренной скорости движения мышки по столу, можно рассчитать скорость ее курсора по формуле. [10]

$DPI = 1600$,

$Rh = 1080$,

$V_{ср.с} = 2$ см/с,

$H = 10{,}79$ д

$$V_{ср.к} = \frac{DPI * V_{ср.с} * H}{R_h} = \frac{1600*2*10.79}{1080} = 32\frac{см}{с}$$

Закон Хика, также известный как закон Хика-Хирша, представляет собой эмпирическое правило, описывающее время принятия решения человеком в зависимости от количества вариантов или альтернатив, из которых ему нужно выбрать. Этот закон был предложен экономистами Р. Дунканом Лэном и Албертом Кайнаном Хиком в 1952 году.

Формула закона Хика имеет вид:

$$T_x = a + b\log_2(n+1),$$

где:

- $(T_x)$ — время принятия решения (время реакции),
- $(a)$ — фиксированное время, не связанное с принятием решений,
- $(b)$ — эмпирическая константа, основанная на времени когнитивного процесса для каждого варианта,
- $(n)$ — количество альтернативных стимулов (вариантов выбора).

Суть закона Хика заключается в том, что время принятия решения увеличивается нелинейно с увеличением количества вариантов. Это связано с тем, что с увеличением числа вариантов происходит увеличение когнитивной нагрузки на человека, что замедляет процесс принятия решения. [7]

Основные черты закона Хика и обоснование константы:

- Нелинейность: Время принятия решения не увеличивается пропорционально количеству вариантов, а возрастает нелинейно.
- Логарифмическая зависимость: Закон Хика основан на логарифмической зависимости времени принятия решения от количества вариантов.
- Эмпирическая константа (*b*): Эта константа отражает время когнитивного процесса, необходимого для обработки каждого варианта.

Теоретическое время при 2 параметрах *a* и *b* соответственно:

$$T_x = 0.25\,с$$

Рассмотрим случай, когда кнопка создания нового элемента находится в верхней части экрана, а список занимает центральную часть экрана. Курсор мышки находится в центре списка.

Закон Фиттса, также известный как закон Фиттса-Шнайдермана, описывает зависимость времени, необходимого для точного движения курсора или указателя (например, мыши), от размера и расстояния до цели. Этот закон был разработан психологом Полом Фиттсом в 1954 году и позднее доработан в соавторстве с Джоном Шнайдерманом. [7]

Формула данного закона выглядит вот так:

$$T_ф = a + b\log_2\left(\frac{D}{W} + 1\right),$$

где

- $(T_D)$ — время движения курсора,
- $(D)$ — расстояние до цели,
- $(W)$ — ширина цели,
- $(a)$ и $(b)$ — эмпирические константы.

Основные положения закона Фиттса:

- *Время движения зависит от расстояния до цели и её размера:* Чем дальше и меньше цель, тем больше времени потребуется для выполнения движения курсора к ней.
- *Логарифмическая зависимость:* Время движения руки к цели в законе Фиттса зависит логарифмически от отношения расстояния до цели к её размеру. Это означает, что увеличение расстояния до цели или уменьшение её размера приводит к увеличению времени движения, но нелинейно.
- *Эмпирические константы (a) и (b):* Эти константы зависят от конкретных условий исследования, таких как характеристики пользователей, интерфейсные особенности и т. д.

Закон Фиттса широко используется в интерфейсном дизайне, особенно в разработке пользовательских интерфейсов компьютерных программ, мобильных приложений, веб-сайтов и других интерактивных систем. Он помогает оптимизировать размеры и размещение элементов интерфейса, чтобы обеспечить удобство и эффективность взаимодействия пользователя с системой. [7]

Чтобы минимизировать расстояние от центра списка до кнопки создания, нужно иметь:

- Margin_кнопки насколько возможно меньше
- Width_кнопки насколько возможно больше

Где Margin_кнопки — отступ кнопки от списка, Width_кнопки — ширина кнопки. [11]

Возьмем высоту элемента списка равную Width_экрана = 1980 px, Height_экрана = 1080 px, ширину кнопки Width_кнопки = 250px и Height_кнопки = 60px, $Margin_{\text{кнопки}} = 0$. Тогда расстояние от курсора в момент начала движения до целевого элемента управления будет в среднем

Расстояние от кнопки до центра списка по горизонтали:

$$x_1 = \frac{Width_{\text{экрана}}}{2} - X_{\text{кнопки}} - Width_{\text{кнопки}} = 600px$$

Расстояние от кнопки до центра списка по вертикали:

$$x_2 = \frac{Height_{\text{списка}}}{2} + Margin_{\text{кнопки}} = 500px$$

$$D = \sqrt{x_1^2 + x_2^2}$$

$$D \approx 782 \text{ px}$$

Теоретическое время при $a = 0$, $b = 0.155$:

$$T_{\text{ф}} = 0.442c$$

Итоговое теоретическое время выполнения действия:

$$T_{\text{мд}} = T_{\text{ф}} + T_{\text{х}} + T_{\text{нкм}} = 0{,}25 + 0{,}442 + 0{,}18 = 0{,}872 \text{ с.}$$

На практике процесс поиска одной книги (Harry Potter) и добавления её в избранное будет выглядеть следующим образом:

Время выполнения действия поиска и добавления в избранное для книги приведено в табл. 5:

Таблица 5.

Время выполнения действия поиска и добавления в избранное для книги

| № | Событие | Время, с |
|---|---|---|
| 1 | Перемещение курсора на 8 см | 0,031*8=0,248 |
| 2 | Нажатие кнопки мыши | 0,18 |
| 3 | Время восприятия человека | 0,15 |
| 4 | Нажатие кнопки мыши | 0,18 |
| 5 | Время восприятия человека | 0,15 |
| 6 | Ввод 13 символов | 0,29*13=3,77 |
| 7 | Перемещение курсора на 2 см | 0,031*2=0,062 |
| 8 | Нажатие кнопки мыши | 0,18 |
| 9 | Время восприятия человеком | 0,15 |
| 10 | Перемещение курсора на 1 см | 0,031 |
| 11 | Нажатие кнопки мыши | 0,18 |
| 12 | Перемещение курсора на 4 см | 0,031*4=0,124 |
| 13 | Нажатие кнопки мыши | 0,18 |
| # | Итого | 5,585 |

Суммарное время выполнения действия составит:

$$5{,}585 + 0{,}872 = 6{,}457\text{с}$$

Таким образом, наибольшее время занимает работа человека-оператора с клавиатурой. Для того, чтобы уменьшить данное время, необходимо добавить всплывающие подсказки при наборе, которые будут эффективно ускорять этот процесс.

Такое эргономическое решение позволит сокращать время по 0,29 секунды за каждый сэкономленный подсказкой символ и упростит взаимодействие пользователя с системой.

## Заключение

Основная задача данного исследования заключалась в изучении и эргономическом анализе программного модуля для подбора книг. Рекомендательная система, на которой основан данный модуль, предлагает пользователю индивидуализированные рекомендации, учитывая его активность и предпочтения. Эргономический анализ в контексте рекомендательных систем является важной областью исследования, поскольку оценивает удобство использования и качество предоставляемых рекомендаций. Он направлен на обеспечение максимальной удовлетворенности пользователей и оптимизацию работы системы.

Результаты эксперимента подтверждают аналитические предположения и формулы, использованные для оценки производительности системы. Наибольшее время выполнения приходится на блок тестирования и визуализации, что требует дополнительного внимания и оптимизации на этапе разработки. При наличии грамотной настройки приложения блок тестирования не будет существенно влиять на время выполнения пользовательских запросов. Однако визуальная составляющая системы зависит от мощности компьютера пользователя и сервера разработчика, а также требований к дизайну системы.

В заключительной части был проведён эргономический анализ пользовательского интерфейса и найдено наиболее затратное по времени место для пользователя, а также предложено решение по устранению данной недоработки.

ЛИТЕРАТУРА

1. Tejeda-Lorente Á., Porcel C., Peis E., Sanz R., Herrera-Viedma E. A quality based recommender system to disseminate information in a university digital library // Inf. Sci. 261, 2014.
2. Григорьев Ю.А., Ревунков Г.И. Г 82 Банки данных: Учеб. для вузов. — М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. — 320 с.
3. Документация Django, https://docs.djangoproject.com/en/3.0/, дата обращения 15.01.2024
4. Burke R. Hybrid web recommender systems // The adaptive web / Lecture Notes In Computer Science. 2007. V. 4321.
5. Гапанюк Ю.Е. Конспект лекций по спецкурсу «Гибридные интеллектуальные информационные системы на основе метаграфового подхода»: Учебно-методическое пособие. — М.: Издательство «Спутник +», 2018. — 53с., ил
6. Программное приложение для определения наилучших условий человеко-компьютерного взаимодействия с использованием законов Фиттса и Хика. Попов А.А. 2016 г. URL: https://cyberleninka.ru/article/n/programmnoe-prilozhenie-dlya-opredeleniya-nailuchshih-usloviy-cheloveko-kompyuternogo-vzaimodeystviya-s-ispolzovaniem-zakonov-fittsa (дата обращения: 28.02.2024). — Текст: электронный.
7. Reza Abbasi-Kesbi, Hamidreza Memarzadeh-Tehran — Technique to estimate human reaction time based on visual perception — 2017 Apr 21 URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5408555/ (дата обращения: 25.01.2024). — Текст: электронный.
8. Горячкин Б.С. Эргономический анализ систем обработки информации и управления //Интернет-журнал Науковедение. — 2017. — Т. 9. — №. 3
9. Время реакции человека URL: https://www.booksite.ru/fulltext/1/001/008/007/002.htm (дата обращения: 23.01.2024). — Текст: электронный.
10. Все о DPI у компьютерной мышки: что это и зачем нужна? А. Андреев 2.05.2019 URL: https://infotechnica.ru/vse-chto-podklyuchaetsya-k-kompyuteru/o-myshkah/vse-o-dpi/ (дата обращения: 21.01.2024). — Текст: электронный.
11. Melville P., Mooney R.J., Nagarajan R. Content-boosted collaborative filtering for improved recommendations // 8th national conf. on Artificial intelligence. Menlo Park, CA, USA, 2002.


Журнал «Современная наука: актуальные проблемы теории и практики»