|  |  |
| --- | --- |
| *Описание: voenmeh* | МИНОБРНАУКИ РОССИИ  федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**  **(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)** |
| БГТУ.СМК-Ф-4.2-К5-01 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | И |  | «Информационные и управляющие системы» |
|  |  | шифр |  | наименование |
| Кафедра |  | И5 |  | «Информационные системы и программная инженерия» |
|  |  | шифр |  | наименование |
| Дисциплина |  | Проблемы человеко-машинного взаимодействия | | |

КУРСОВАЯ РАБОТА

на тему

|  |
| --- |
| Построение способов взаимодействия пользователей |
| с разрабатываемым программно-аппаратным |
| комплексом |

Вариант 25

Программа чтения электронных книг для планшета/ноутбука-трансформера

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | | | |  | И9М31 |
| Масленников В.М. | | | | | |
| Фамилия И.О. | | | | | |
| **РУКОВОДИТЕЛЬ** | | | | | |
| Гущин А.Н. | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| Оценка |  | | | |  |
| «\_\_\_\_\_» |  | | | | 2018 г. |

САНКТ-ПЕТЕРБУРГ

2018 г.

Содержание

Введение3

1. Выбор целевого программно-аппаратного комплекса4
   1. Целевая аудитория4
   2. Программно-аппаратная платформа6
2. Разработка пользовательского интерфейса11
   1. Анализ способов взаимодействия с пользователем в аналогичных системах11
   2. Основные требования к пользовательскому интерфейсу14
   3. Макетирование пользовательского интерфейса18
   4. Оценка пользовательского интерфейса21

Заключение25

Список использованных источников26

Введение

Быстрорастущий рынок мобильных приложений продолжает завоевывать самые разные сферы нашей жизни. Люди активно используют мобильные уст в повседневной жизни, что, несомненно, увеличивает спрос на самые разные приложения для мобильных телефонов. Мобильные устройства – это та технология, которую люди все время держат под рукой, так как с их помощью можно крайне быстро получить достоверные сведения. Одними из поставщиков таких сведений является литература. На сегодняшний день в сети существуют тысячи онлайн-библиотек, платных и бесплатных, предоставляющих книги в разных форматах и на разные темы. Для сравнения: Российская государственная библиотека имеет около 40 миллионов единиц хранения.

Основные преимущества электронной версии книги над её печатным изданием:

1. Компактность и портативность. На одном устройстве можно хранить огромное количество произведений, которые всегда будут под рукой.
2. Возможность не только читать, но и прослушать книгу в mp3 формате.
3. Относительно низкая стоимость, многие тексты в электронном виде бесплатны или дешевле, чем в бумажном.
4. Доступность. При наличии подключения к Интернету тексты можно скачать в любое удобное время, воспользовавшись соответствующей онлайн-библиотекой.
5. Простота публикации. Текст проще и дешевле опубликовать в электронном виде, чем в бумажном.
6. Экологичность. Чтобы читать электронную книгу бумага не нужна, а значит и деревья не надо вырубать.

Недостатки есть у всего и электронные книги не исключение:

1. Продолжительность чтения напрямую зависит от заряда батареи устройства.
2. Наличие электронного устройства, способного выполнять задачи электронной книги.
   1. Выбор целевого программно-аппаратного комплекса

Для того чтобы выбрать целевой программно-аппаратный комплекс, необходимо дать ему определение. Программно-аппаратный комплекс — это набор технических и программных средств, работающих совместно для выполнения одной или нескольких сходных задач [1]. В нашем случае, под задачами понимается предоставление пользователю возможности чтения книг на планшете или ноутбуке-трансформере.

* 1. Целевая аудитория

Наличие целевой аудитории у продукта позволяет сконцентрироваться на конкретной группе потребителей рынка и создать для них идеальный товар. Как правило, целевой аудиторией является та группа людей, которой нужен продукт и которой интересны преимущества, предлагаемые продуктом. При оценке целевой аудитории следует, прежде всего, выяснить географию будущих пользователей. Предположим, что первые версии программы будут рассчитаны на русскоязычных пользователей, с возможностью в дальнейшем перевести интерфейс программы на другие языки. Кроме того необходимо учитывать направление правописания справа налево в некоторых языках (арабский, иврит).

Для того чтобы получить больше информации о целевой аудитории можно воспользоваться статистикой всероссийского центра изучения общественного мнения (ВЦИОМ), который предоставляет данные исследования, посвященного теме чтения. На конец 2017 года, до 15 % сократилась доля граждан, признающихся, что они практически не обращаются к литературным изданиям. До 55 % россиян называют конкретное число книг, прочитанных за последние три месяца, в среднем – 6,49 книг [2]. В основном, книги читают люди в возрасте от 18 до 70 лет и выше, при этом число читающих женщин больше. Больше всего из опрошенных, регулярно читают своим чадам детскую литературу - сказки, стихи, повести и рассказы. Еще около 30 процентов отдают предпочтение историческим романам. 27 % регулярно штудируют полезную литературу – различные справочники, книги по ведению домашнего хозяйства, ухода за растениями и приусадебным участком. Также не стоит обделять вниманием образовательные учреждения. Но сейчас относительно небольшое их количество использует электронные учебники. В ходе опроса удалось выяснить, что в настоящее время использование планшетов в российских школах скорее редкость: 81% учителей, 90% родителей и 88% учеников признались, что пока подобных уроков в их школах нет [16].

Люди, как правило, читают либо бумажные книги, либо электронные. Электронные книги можно читать на множестве различных типов мобильных устройств. Из этого множества нас интересуют ноутбуки-трансформеры и планшеты. На рисунке 1 показано процентное соотношение продаваемых планшетов, мобильных устройств и ноутбуков на российском рынке за 2018 год.

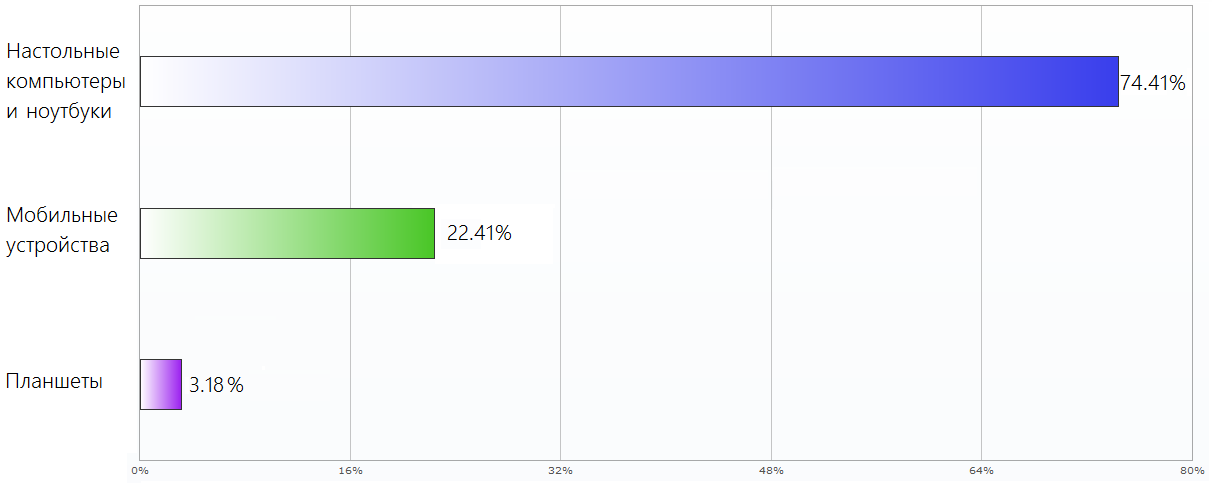


Рисунок 1 — Динамика рынка устройств в России за 2018 год.

Данная статистика базируется на данных полученных более чем от 2 миллионов веб-сайтов [3]. Видно, что наибольшая популярность сохраняется за ноутбуками и настольными ПК. Планшеты же составляют относительно небольшую долю рынка. Процентное соотношение используемых операционных систем в планшетах представлено на рисунке 2.

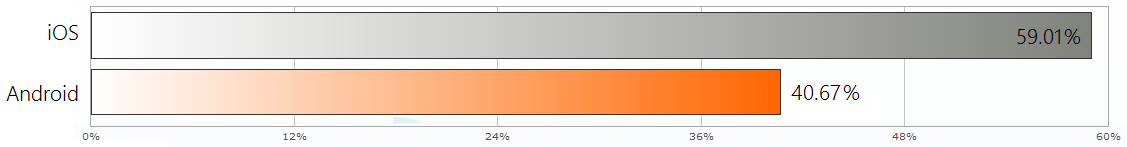


Рисунок 2 — Доля операционных систем в планшетах

Процентное соотношение используемых операционных систем в настольных компьютерах представлено на рисунке 3. Видно, что в планшетах, доминирующими являются две операционные системы: iOS (версии 11.4 и 12.0) и Android (начиная с версии 4.4), их доля составляет 59,01% и 40,67% соответственно. На рынке настольных компьютеров и ноутбуков доминирующей является одна система – Windows (Windows 7 – 48%, Windows 10 – 35%, Windows 8.1 – 9%).

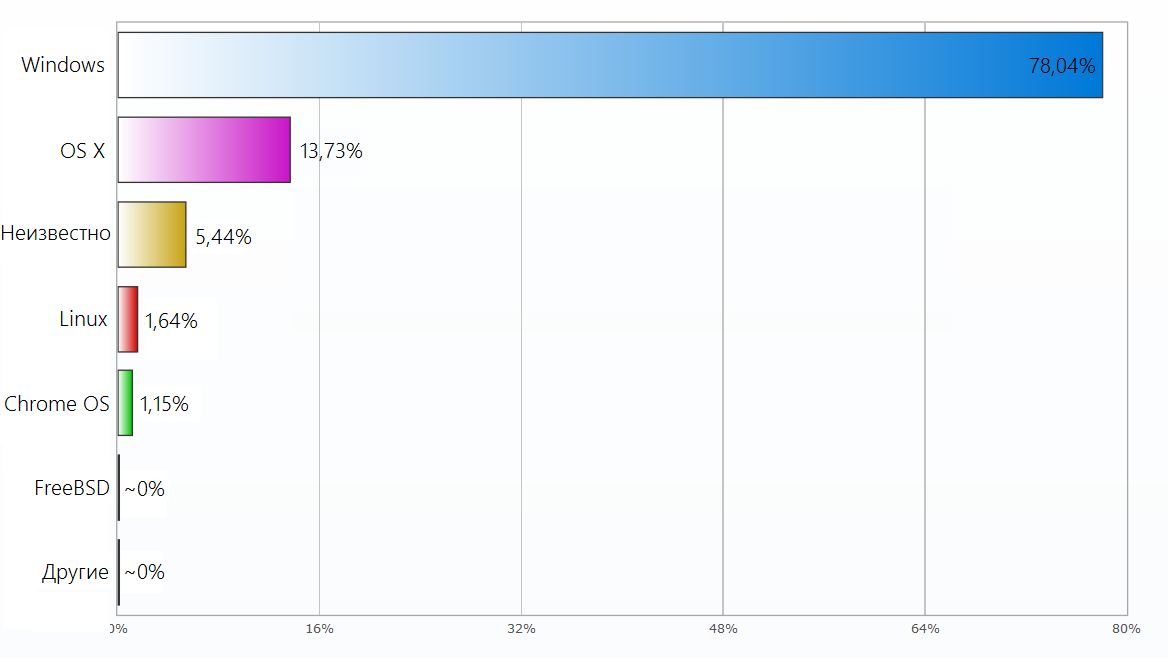


Рисунок 3 — Доля операционных систем в настольных компьютерах и ноутбуках.

Таким образом, целевой аудиторией программы являются русскоязычные мужчины и женщины в возрасте от 18 лет использующие ноутбуки-трансформеры с операционной системой Windows 7 и выше, OS X, Linux, Chrome OS, и планшеты с операционной системой iOS 11.4 и выше, и Android 4.4 и выше.

* 1. Программно-аппаратная платформа

Компьютерная платформа — в общем смысле, это любая существующая среда выполнения, в которой должен выполняться вновь разрабатываемый фрагмент программного обеспечения или объектный модуль с учётом накладываемых этой средой ограничений и предоставляемых возможностей [4].

Аппаратная платформа представляет собой нижний слой многоуровневой организации вычислительной системы (аппаратура, операционная система, прикладное программное обеспечение), на который опираются операционная система и прикладное программное обеспечение. Аппаратные платформы отличаются друг от друга архитектурой центрального процессора и используемыми шинами связи между элементами. Каждой аппаратной платформе соответствуют совместимые с ней операционные системы и прикладные программы, которые могут на ней запускаться.

Программная платформа представляет собой общую организацию исполнения прикладных программ, задавая, например, порядок запуска программы, схему использования ею адресного пространства, зафиксированные в архитектуре операционной системы плюс API на уровне операционной системы.

Для улучшения качества программы, ее безопасности и увеличения ее возможностей, мы будем использовать привязку к конкретным программным платформам. Однако существует иной подход при разработке программного обеспечения, использующий сторонние инструменты и библиотеки. Способность программного обеспечения работать более чем на одной аппаратной платформе или операционной системе называется кроссплатформенностью. Она достигается благодаря кроссплатформенным стандартизованным библиотекам среды выполнения. В частности, стандартом стала библиотека языка Си. Из крупных кроссплатформенных библиотек — Qt, GTK+, FLTK, STL, Boost, OpenGL, SDL, OpenAL, OpenCL. Также, кроссплатформенными можно назвать большинство современных высокоуровневых языков программирования. Например, C, С++, FreePascal, FreeBASIC, PureBasic — кроссплатформенные языки на уровне компиляции, то есть для этих языков есть компиляторы под различные платформы. Это позволяет — при надлежащем качестве кода — не переписывать основной движок программы, меняются только особые системозависимые части.

Кроме того, существуют интерпретируемые языки, одним из основных преимуществ которых является кроссплатформенность [5]. Для запуска программы на интерпретируемом языке программирования, необходима среда выполнения. Среда выполнения — вычислительное окружение, необходимое для выполнения компьютерной программы и доступное во время выполнения компьютерной программы. Взаимодействие со средой выполнения для интерпретируемых языков программирования реализуется непосредственно в интерпретаторе, обеспечивающим взаимодействие конструкций языка с окружением, в котором он запущен. Основным недостатком интерпретируемых языков программирования является более медленное выполнение программы [6, с. 215] по сравнению с выполнением программы, предварительно скомпилированной в машинный код. Например, выполнение PHP и Python может оказаться в более чем 100 раз медленнее, чем C++ [7, c. 585]. Трансляция в байт-код и JIT-компиляция не решают этой проблемы полностью. Дополнительный слой интерпретатора или виртуальной машины замедляет выполнение программы и может требовать больше ресурсов.

Кроме того, на разных операционных системах — независимо от того, как технически достигнута работа в них — стандартные элементы интерфейса имеют разные размеры. Поэтому простое жёсткое позиционирование элементов интерфейса невозможно — под другой операционной системой они могут налезать друг на друга. Существует несколько подходов для решения этой проблемы:

1. Единый стиль, общий для всех ОС. Программы выглядят одинаково под всеми ОС. Так работают интерфейсные библиотеки Java наподобие Swing. Однако программе приходится иметь свои экранные шрифты, и их стиль может отличаться от стиля шрифтов операционной системы.
2. Самоадаптирующийся интерфейс, подстраивающий сетку под реальные размеры элементов управления. Типичные примеры — Qt, wxWidgets, XUL. Однако, чтобы собрать самоадаптирующуюся сетку, требуется квалифицированный программист. Кроме того затруднена плотная компоновка элементов.

В любом случае, под другими ОС требуется хотя бы минимальное тестирование, так как возможны ошибки компоновки.

Для разработки программ под Microsoft Windows, используются комплекты разработки программного обеспечения от Microsoft, которые содержат файлы заголовков, библиотеки, примеры, документацию и инструменты разработки. SDK для Windows доступен бесплатно на Microsoft Download Center.

Для разработки программ под macOS (она же OS X) можно использовать один из пяти основных API — Cocoa, Carbon, Toolbox (для работы старых приложений Mac OS 9), POSIX и Java. Остановимся подробнее на Cocoa. Приложения, использующие Cocoa, обычно разрабатываются с помощью среды разработки Apple Xcode (в прошлом называвшегося Project Builder) и Interface Builder с использованием языков программирования: C, Objective-C и Swift. Cocoa состоит в основном из двух библиотек объектов Objective-C, называемых фреймворками. Фреймворки — это примерно то же, что и динамические библиотеки [8, c. 261]. Они представляют собой скомпилированные объекты, загружаемые в адресное пространство программы во время исполнения, но помимо этого фреймворки включают ресурсы, заголовочные файлы и документацию. Cocoa также включает систему контроля версий, предупреждающую проблемы, встречающиеся в Microsoft Windows (так называемый «DLL hell»). Ключевой элемент архитектуры Cocoa — это модель представлений (views), позволяющая программисту рисовать всё, что угодно, используя команды языка, похожего на PostScript.

Для разработки программ под Linux можно использовать кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++ — Qt. Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем, например в Linux, Mac OS X, Windows, путём простой компиляции программы для каждой системы без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML.

Для разработки программ под Chrome OS можно использовать язык программирования JavaScript, либо использовать инструменты, позволяющие эффективно исполнять C, C++ код в браузере, такие как Native Client, либо использовать инструменты транслирования программ на языке C++ в программы на языке JavaScript, такие как Emscripten.

Для разработки программ под iOS используется комплект средств разработки iOS SDK, который включает в себя среду разработки Apple Xcode. Разработка ведется с использованием языков программирования: C, Objective-C и Swift.

Для разработки программ под Android используется универсальное средство разработки мобильных приложений для операционной системы Android — Android SDK [9], который имеет широкие функциональные возможности, позволяющие запускать тестирование и отладку исходных кодов, оценивать работу приложения в режиме совместимости с различными версиями ОС Android и наблюдать результат в реальном времени (опционально). Поддерживает большое количество мобильных устройств, среди которых выделяют: мобильные телефоны, планшетные компьютеры, умные очки (в том числе Google Glass), современные автомобили с бортовыми компьютерами на ОС Андроид, телевизоры с расширенными функциональными возможностями, особые виды наручных часов и многие другие мобильные гаджеты, габаритные технические приспособления. Разработка ведется на языке Java.

К ограничениям аппаратной платформы можно отнести размер экрана устройства, объем оперативной памяти, время автономной работы, способы ввода-вывода информации. К ограничениям программной платформы можно отнести возможность программной регулировки яркости экрана, перепрограммирование кнопок, наличие библиотек для работы с популярными форматами электронных книг.

Определив средства разработки под различные программные платформы, можно перейти к функциональным особенностям приложения.

1. Разработка пользовательского интерфейса

Пользовательский интерфейс — интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы [10]. Возможные варианты ввода информации — жестовый, голосовой, или смешанные варианты. Средства ввода должны быть удобными и практичными, расположенными и скомпонованными разумно и понятно, соответствовать физиологии человека, не должны приводить к негативным последствиям для организма пользователя.

* 1. Анализ способов взаимодействия с пользователем в аналогичных системах

Планшет (электронный планшет) — понятие, включающее различные типы электронных устройств с сенсорным экраном, позволяющим управлять компьютерными программами, через прикосновение пальцами к объектам программы на экране.

Ноутбук-трансформер — вид мобильных персональных компьютеров типа ноутбук с сенсорным экраном и c поворотным дисплеем, а возможно и со съёмной клавиатурой.

Общим принципом взаимодействия с пользователем у ноутбуков-трансформеров и электронных планшетов является способ взаимодействия через сенсорный экран, а также возможностью вывода информации с помощью звука. Существует множество различных программ для чтения электронных книг, для обоих типов устройств. Рассмотрим интерфейс некоторых из них.

Calibre — программа, хранящая книги электронной формы, в ней имеется удобная система поиска требуемых пользователю изданий и коллекций книг. С помощью Calibre можно производить конвертацию форматов электронных книг. Кроме того, программа имеет большое количество настроек визуального оформления читаемой книги, конвертирование с учетом огромного количества параметров текста. Внешний вид программы представлен на рисунках 4 и 5.

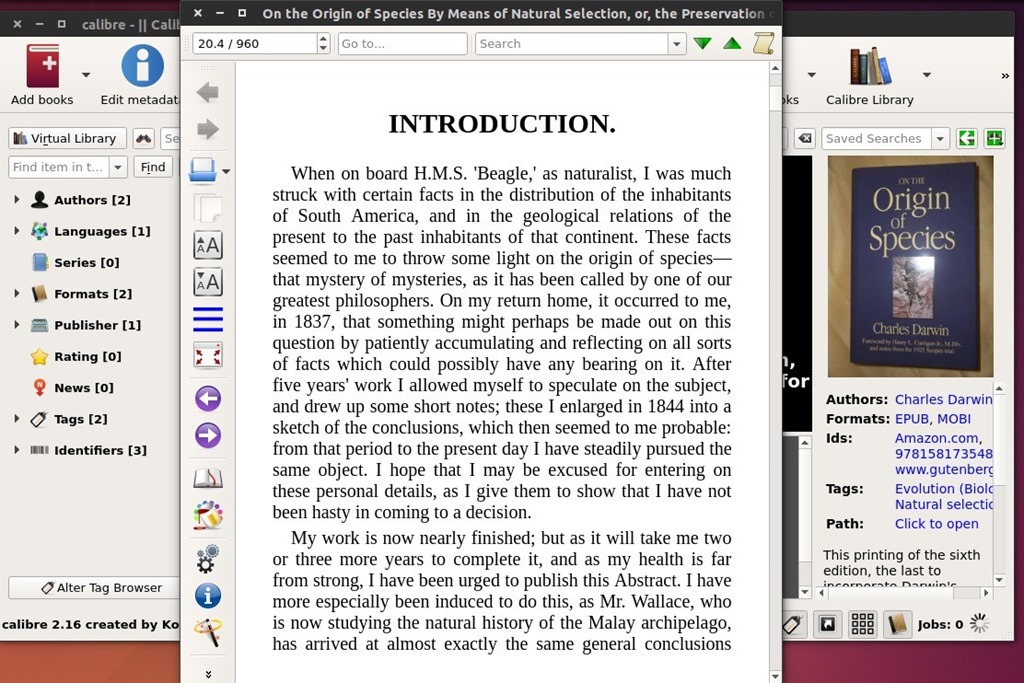


Рисунок 4 — Режим чтения в программе Calibre.

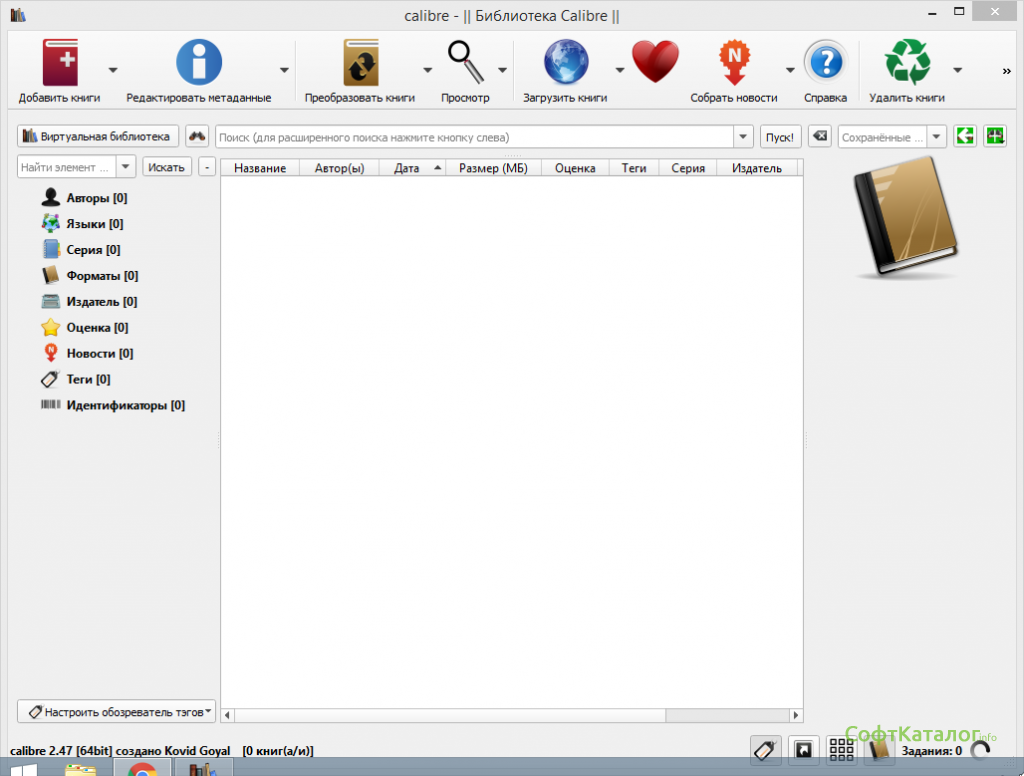


Рисунок 5 — Главное окно программы Calibre.

Программа Cool Reader имитирует привычный вид раскрытой книги с плавным перелистыванием и масштабированием размера шрифта. Присутствуют широкие настройки отображения книги для чтения, как в виде непрерывного текста, так и в виде обычного перелистывания страниц. Количество доступных пользователю настроек очень сильно уступает таким монстрам как Calibre и они в основном касаются визуального оформления. Среди дополнительных возможностей — добавление книг в архивах, запись книг в mp3 для прослушивания в плеере и чтение различными голосами. Все типичные возможности типа закладок, поиска по тексту, запоминания списка ранее открытых файлов и поддержки русского языка, присутствуют. Так же есть возможность доступа к онлайновым каталогам, для которых понадобится отдельная авторизация и доступ к сети «Интернет». На рисунке 6 представлен интерфейс программы Cool Reader.

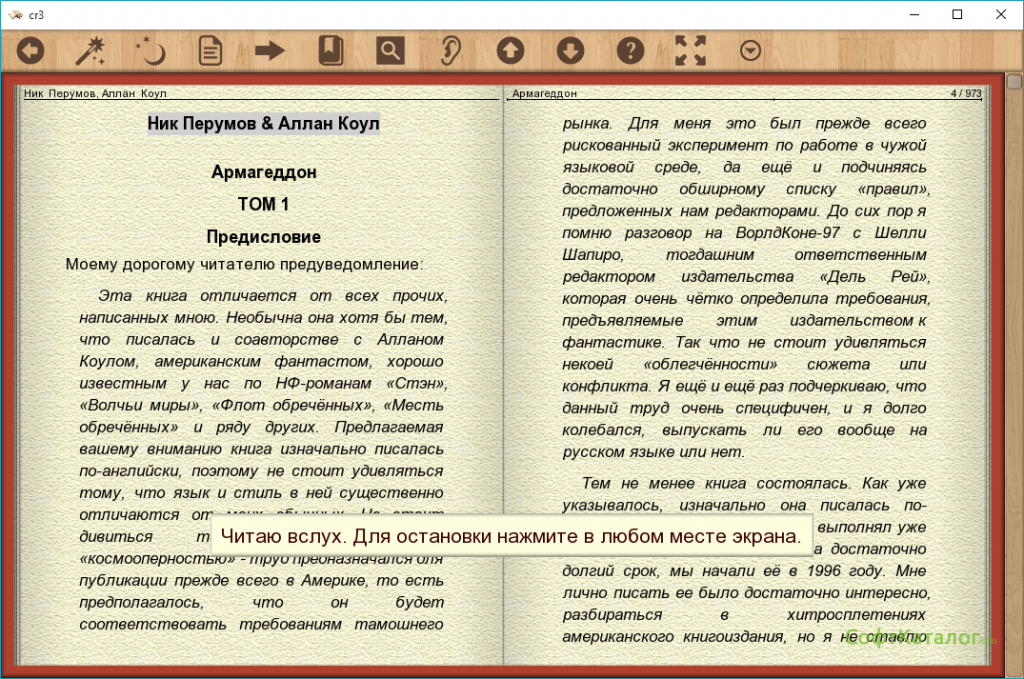


Рисунок 6 — Пользовательский интерфейс программы Cool Reader.

Kindle Cloud Reader — программа для чтения электронных книг от Amazon Kindle, позволяющая читать электронные книги Amazon из браузера или Chromebook. Kindle Cloud Reader – гибкое средство для чтения с опциями изменения размера шрифта и многими другими возможностями. Позволяет изменять внешний вид, настраивать шрифты, цвет фона и текста читаемой книги. Интерфейс программы представлен на рисунке 7.

Из рассмотренных примеров можно сделать некоторые выводы. Например, в программе необходимо предусмотреть гибкую настройку внешнего вида читаемой книги, добавить возможность воспроизведения книги электронным голосом.

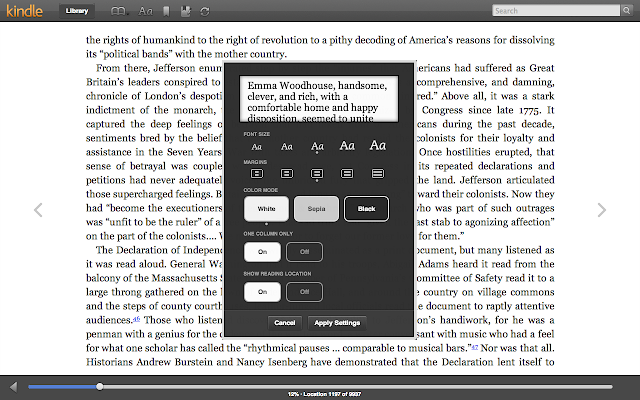


Рисунок 7 — Пользовательский интерфейс программы Kindle Cloud Reader

Во всех рассмотренных программах присутствует графический пользовательский интерфейс, реализующий стиль взаимодействия пользователя с компьютером — WIMP. Присутствует окно программы, в котором существует строка меню, навигационное меню, рабочая область. Взаимодействие с пользователем происходит с помощью дополнительных диалоговых окон. Диалоговые окна могут содержать текстовые поля, кнопки, списки, флажки и другие элементы управления. Также возможно применение SILK интерфейса, в котором взаимодействие осуществляется с помощью голосовых команд пользователя.

* 1. Основные требования к пользовательскому интерфейсу

Основные требования к пользовательскому интерфейсу:

* функциональность (соответствие задачам пользователя);
* соответствие технологии;
* понятность и логичность;
* обеспечение высокой скорости работы пользователя;
* обеспечение защиты от человеческих ошибок;
* быстрое обучение пользователя;
* субъективное удовлетворение пользователя.

Основные требования к функциональности:

* возможность отображения всех популярных форматов электронных книг: cbz, cbr, cbc, doc, docx, epub, fb2, html, lit, lrf, а также pdb, pdf, pml, pmlz, prc, rar, rb, rtf, zip;
* возможность покупать или загружать бесплатно книги из сторонних источников;
* возможность производить поиск книги по записи как в одной книге, так и в целой библиотеке книг, поиск книг по названию;
* интерфейс на русском языке;
* поддержка горячих клавиш;
* возможность настройки внешнего вида программы и книги;
* возможность работы с закладками, а также инструменты выделения и копирования текста;
* регулировка масштаба открытой книги.

Требование функциональности пользовательского интерфейса, заключается в способности интерфейса выполнять набор задач, определенных в его внешнем описании и удовлетворяющих заданным потребностям пользователей. Описание требований к интерфейсу приведено выше. Таким образом, необходимо обеспечить выполнение этих требований на выбранной аппаратно-программной платформе.

Требование соответствия технологии заключается в соответствии разрабатываемого пользовательского интерфейса выбранной технологии разработки пользовательского интерфейса. Технология разработки программного обеспечения — это совокупность процессов и методов создания программного продукта. Программирование, несмотря на интеллектуальность и творческий характер этой деятельности, нуждается в организации и регламентировании, наборе соглашений и правил. Обобщённый термин, применимый к созданию программных средств, обозначают как «разработка» или «конструирование». Справедлива формула: разработка = анализ + проектирование + программирование (кодирование) + тестирование + отладка [11, с. 7]. Это же правило можно применить и к пользовательскому интерфейсу.

Требование понятности и логичности заключается в верности отражения фактов действительности и их причинно-следственных связей относительно жизненного опыта пользователя. Как правило, пользователь при взаимодействии с программой, ожидает получить от нее какой-то результат. Если пользователь нажимает какую-то кнопку, он ожидает, что произойдет какое-то событие. Соответствие ожидания пользователя результатам и характеризует понятность и логичность.

Требование обеспечения высокой скорости работы пользователя заключается в создании интерфейса, в котором мыслительные действия, направленные на поиск решения задачи, сводятся к минимуму. С этой точки зрения, пользователя можно охарактеризовать скоростью познавательной (когнитивной) обработки информации - когнитивной способности, которая может быть определена как время, необходимое человеку для решения умственной задачи. Этот навык связан со скоростью, с которой человек улавливает информацию и реагирует на неё, как с помощью зрения (через буквы и цифры), так и с помощью слуха (речь) или движения. Т.е. скорость обработки информации - это время между получением стимула и ответом на него. Скорость обработки информации подразумевает способность свободно выполнять лёгкие или знакомые задания [12]. Пользователь должен как можно меньше думать о том, где же находится нужный ему элемент. Некоторые примеры неправильного построения интерфейса, мешающие быстрой работе пользователя: перегруженные меню, неправильная группировка пунктов меню, некорректные заголовки пунктов меню, неправильная группировка информации, неинформативное содержание, отсутствие отступов, несочетаемые шрифты, отвлекающие изображения или элементы.

Иногда психологические и психофизиологические характеристики человека не соответствуют уровню сложности решаемых задач или проблем. Характеристики, возникающие при взаимодействии человека и технических систем, часто называют «человеческий фактор». Ошибки, называемые проявлением человеческого фактора, как правило, непреднамеренны: человек выполняет ошибочные действия, расценивая их как верные или наиболее подходящие. Требование обеспечения защиты от человеческих ошибок заключается в способности интерфейса предугадывать нежелательные действия пользователя и предупреждать пользователя о последствиях таких действий. В нашем случае, ошибочным может быть случайное удаление книги из библиотеки, поэтому при всяком удалении книги, необходимо запрашивать подтверждение данного действия у пользователя.

Требование быстрого обучения пользователя заключается в дополнительных функциональных особенностях интерфейса, направленных на обеспечение принудительного обучения пользователя. Сюда могут входить различные всплывающие подсказки, диалоговые окна с информацией о том, как получить доступ к какому-либо элементу управления, дополнительное окно справки и поддержки. Также необходимо предусмотреть несколько режимов работы программы. Например, режим новичка, режим пользователя, режим опытного пользователя. Режимы должны отличаться детализированностью и количеством подсказок.

Требование субъективного удовлетворения пользователя заключается в обеспечении пользователю положительного опыта работы с программой. Необходимо чтобы у пользователя остались хорошие впечатления и ему захотелось порекомендовать данную программу своим знакомым. Положительный опыт работы с программой для чтения книг может быть достигнут:

* избеганием ярких, холодных цветов и наличием умеренных контрастов;
* использованием специально спроектированных шрифтов;
* использованием модульных сеток, то есть привязки всех объектов интерфейса к линиям (лучше узлам) воображаемой сетки;
* привязкой всех размеров и координат (как минимум пропорций диалоговых окон) к золотому сечению.
  1. Макетирование пользовательского интерфейса

В проектировании графического интерфейса пользователя нет устоявшихся стандартов, есть отдельные рекомендации, приемы, законы дизайна, традиции, условия эксплуатирования. При этом важной частью этого процесса является прототипирование, или макетирование, т.е. создание макета или прототипа будущей системы. Большинство современных сред программирования позволяет создавать некое подобие макетов, но это сопряжено с определёнными знаниями конкретной среды и конкретного языка. При создании большого проекта, пользовательским интерфейсом, как правило, занимается отдельный человек, который не обязательно участвует в программировании.

Польза от создания макета выражается в следующем:

* возможность рассмотреть будущий интерфейс программы с реальным взаимодействием его частей, не разрабатывая программу целиком, что уменьшает расходы по изменению интерфейса (т.к. чем раньше интерфейс будет приведен к своему окончательному виду, тем дешевле будет итоговый продукт);
* более наглядная демонстрация будущих возможностей конечному пользователю или заказчику;
* проверка практичности, понятности будущего графического интерфейса пользователя;
* тестирование нестандартных, новых подходов к интерфейсу.

На рисунке 8 изображено главное окно программы. Окно разделено на 3 части, размер которых может динамически изменяться. В первой части представлено меню программы, во второй части представлена навигация по книгам и библиотекам, в третьей части показана открытая в данный момент книга. Первые две части могут сворачиваться, для расширения поля обзора открытой книги. Все части могут оснащаться полосами прокрутки, поэтому отсутствует ограничение на количество записей. Кроме того, для быстрой прокрутки в меню навигации по книгам, присутствуют элементы управления для быстрого перехода на несколько строк вперед. Рабочая область, в которой отображается книга, оснащена регулировкой масштаба книги, а также индикатором текущей отрытой страницы.

Для обеспечения взаимодействия с помощью тачскрина, все кнопки сделаны достаточно большого размера. В режиме менеджера книг (рисунок 9), можно добавлять, удалять, редактировать название и информацию о книгах. Доступна возможность импорта и экспорта книг в различные форматы. В меню настроек (рисунок 10) представлена возможность менять цвет фона и текста книги, включать «ночной» режим. Важную роль необходимо придать анимации переходов и цветам интерфейса. Например, чтобы показать текущую выбранную книгу в списке, можно поменять фон строки, в которой расположена запись об этой книге. То же самое можно применить и к пунктам различных меню.

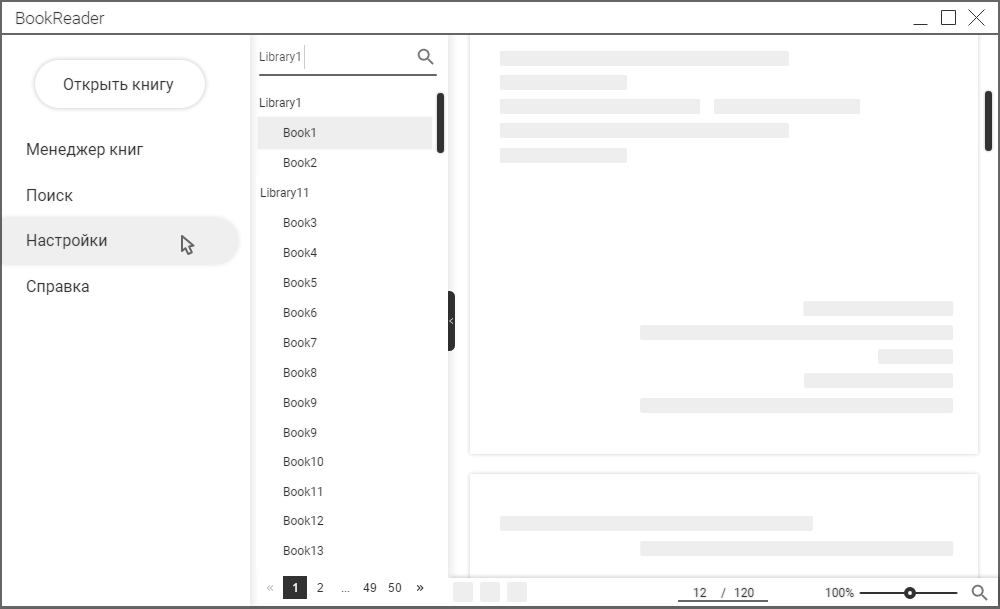


Рисунок 8 — Главное окно программы

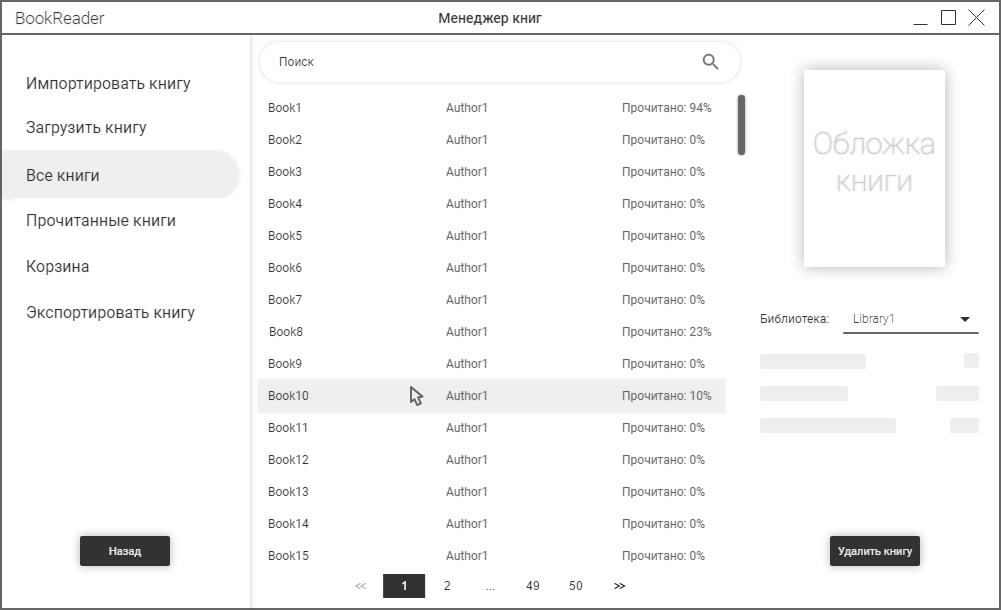


Рисунок 9 — Окно редактирования коллекции книг

Также, можно вынести наиболее важную информацию о книге непосредственно в список книг. Это поможет пользователю знать всю необходимую информацию без дополнительных обращений к интерфейсу.

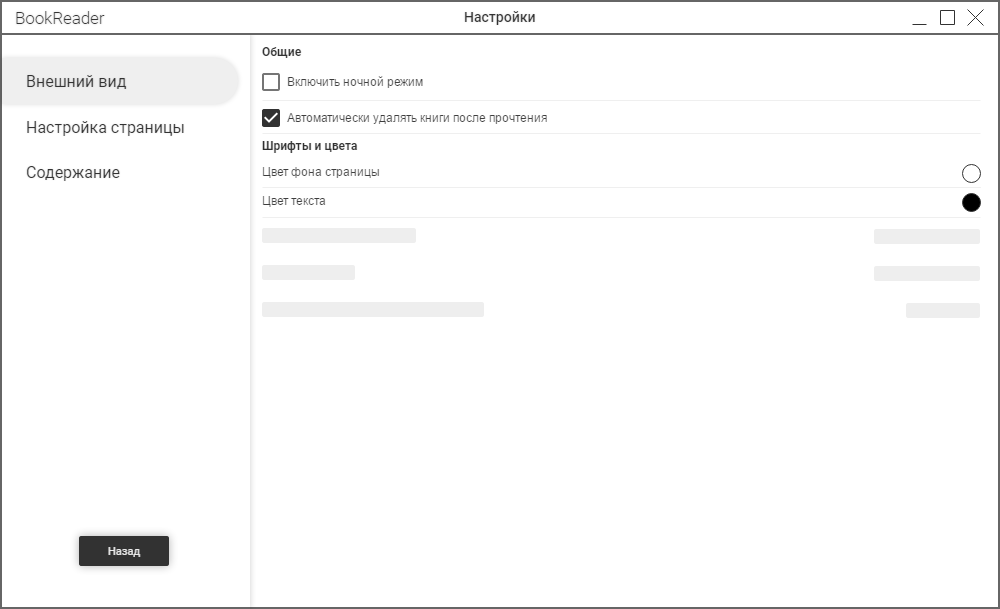


Рисунок 10 — Окно настроек программы

* 1. Оценка пользовательского интерфейса

В процессе проектировании программного обеспечения для большинства современных операционных систем с графическим интерфейсом пользователя и создании программного обеспечения на базе таких операционных систем, неизбежно встает вопрос о проектировании и разработки качественного пользовательского интерфейса. Очевидно, что пользовательский интерфейс напрямую зависит от решаемых программным обеспечением задач, входных и выходных данных. Однако при этом существуют значительная свобода в том, в каком виде все эти данные будут представлены пользователю. На процесс проектирования пользовательского интерфейса, пожалуй, набольшее влияние оказывают субъективные представления проектировщика о понятности, удобстве и красоте. Поэтому, большое значение имеет проблема оценки качества пользовательского интерфейса. Проводя такие оценки на ранних этапах процесса проектирования можно избежать большого числа ошибок, просчетов, неприятия программного обеспечения конечными пользователями. Существует целый ряд подходов позволяющих оценить качество пользовательского интерфейса. В целом все методы можно разбить на две большие группы: методы непосредственно тестирования интерфейса группой пользователей и методы без такового тестирования, основанные на формальных расчетах. Выбор группы методов зависит, главным образом, от того, насколько осуществимо непосредственное тестирование на той или иной стадии выполнение проекта и отведенного на такое тестирование времени и бюджета. Важно учитывать не только стоимость самого проектирования и разработки качественного и удобного пользовательского интерфейса, но и возможных финансовых потерь, которые неизбежны, если интерфейс недостаточно проработан или неудобен в использовании.

Хотя оценка качества пользовательского интерфейса процесс достаточно субъективный и трудно формализуемый [13, c. 5], можно с уверенностью утверждать, что хороший интерфейс должен обеспечивать эффективную и производительную работу пользователя.  Существует также и ряд критериев, которым должен удовлетворять качественный интерфейс:

* лучше тот интерфейс, при котором время выполнения задачи меньше;
* лучше тот интерфейс, в котором число непроизвольных ошибок пользователя меньше;
* неоднозначность в понимании интерфейса должна быть минимальна (это способствует самообучению пользователей и делает их поведение предсказуемым);
* необходима высокая стандартизация интерфейса (она облегчает обучение пользователей);
* объем вводимой пользователем информации должен стремиться к минимуму (одни и те же данные не должны вводиться несколько раз)
* простота и визуальная привлекательность (удобство использования не менее важно, чем функциональность).

Наиболее распространенные методы оценки качества пользовательского интерфейса представлены на рисунке 11 [14].

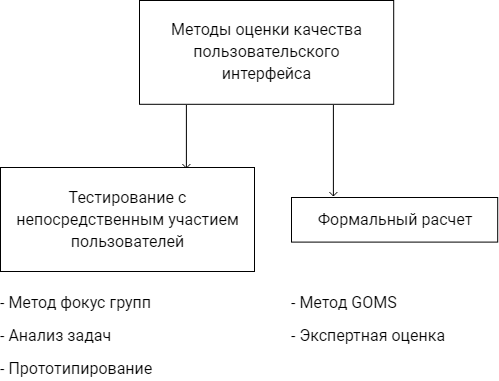


Рисунок 11 — Методы оценки интерфейса

Почти все методы оценки интерфейса требуют наличие группы из нескольких человек. Это позволит оценить интерфейс более качественно, так как пользователи могут быть максимально приближенными к целевой аудитории. С другой стороны данный подход более затратный, как по времени, так и по стоимости. Для оценки качества интерфейса будем использовать метод GOMS.

GOMS это семейство методов позволяющих провести моделирование выполнения той или иной задачи пользователем и на основе такой модели оценить качество интерфейса [15]. Идея метода заключается в том, что все действия пользователя можно представить как набор типовых составляющих (например, нажать ту или иную кнопку на клавиатуре или передвинуть мышь). Для этих типовых составляющих можно провести измерения времени их выполнения (на большом числе пользователей) и получить статистические оценки времени выполнения того или иного элементарного действия. Оценка качества интерфейса заключается в разложение выполняемой задачи на типовые составляющие, и вычисление времени, которое будет в среднем затрачиваться пользователем на выполнение этой задачи. В данном методе каждая цель или задача, которую хочет достичь пользователь с помощью интерфейса, состоит их набора методов, которые в свою очередь построены из операторов. Преимущества метода:

* простота и удобство расчетов;
* отсутствие параметров в модели позволяет проводить оценочные сравнение двух разных вариантов интерфейса;
* дает прогноз времени работы пользователя с данным вариантом интерфейса;
* модель не требует создания рабочего прототипа;
* анализ по этой модели может быть автоматизирован.

Наиболее заметными являются следующие ограничения:

* метод ориентирован на средних пользователей, и не учитывает особенностей работы новичков и специалистов, а также индивидуальных различий пользователей;
* метод не учитывает возникновение случайных ошибок в работе;
* модель не учитывает, что в процессе работы происходит научение, а при простое – забывание;
* модель не учитывает насколько представляемая интерфейсом информация сложна для понимания пользователем;
* модель не учитывает насколько интерфейс отвечает требованиям пользователей и их ожиданиям.

Сравним оценки качества описанного выше пользовательского интерфейса и интерфейса программы чтения электронных книг Calibre. Используем следующие операторы: П — перенести руку (в случае сенсорного экрана) или передвинуть указатель мыши в нужную точку (1,1 сек.); Н — сделать тап или нажать левой кнопкой мыши (0,1 сек); А — анализ дальнейших действий (1,2 сек.). Пусть перед пользователем стоит задача загрузки книги с дискового устройства и ее немедленное открытие. В разработанном интерфейсе необходимо проделать следующие операции П — Н — А — П — Н — П — Н, всего затрачено 4,8 секунды. В интерфейсе программы для чтения электронных книг Calibre: П — Н — А — П — Н — П — Н — А — П — Н, всего затрачено 7,3 секунды. Другой задачей может являться открытие окна с настройками программы. Для этого в Calibre пользователю необходимо выполнить следующую последовательность действий: П — Н — А — П — Н, что составляет 3,6 секунды. В разработанном интерфейсе, решение данной задачи сводится к следующим действиям: П — Н, что составляет 1,2 секунды. Таким образом, с точки зрения метода оценки качества интерфейса GOMS, разработанный пользовательский интерфейс в рассмотренных выше задачах, оказался более продуктивным, чем интерфейс программы для чтения электронных книг — Calibre.

Заключение

В ходе выполнения курсовой работы была сформулирована целевая аудитория разрабатываемого программно-аппаратного комплекса, описана программно-аппаратная платформа, ее возможности и ограничения, проанализированы способы взаимодействия с пользователем в аналогичных системах. Сформулированы требования к пользовательскому интерфейсу, произведено предварительное проектирование пользовательского интерфейса с его макетированием. Произведена оценка полученного интерфейса в сравнении с аналогами. В результате выяснилось, что полученный интерфейс в некоторых задачах превосходит по производительности интерфейсы аналогичных программ, в частности программы для чтения электронных книг — Calibre.

Список использованных источников

1. Словари и энциклопедии на Академике [Электронный ресурс]. URL: https://dic.academic.ru/dic.nsf/ruwiki/166710, дата обращения (11.11.2018)
2. Всероссийский центр изучения общественного мнения [Электронный ресурс]. URL: https://wciom.ru/index.php?id=236&uid=116580, дата обращения (11.11.2018)
3. Инструмент статистики пользователей [Электронный ресурс]. URL: http://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/russian-federation/#monthly-201710-201810, дата обращения (11.11.2018)
4. Словарь компьютерных терминов [Электронный ресурс]. URL: http://foldoc.org/platform, дата обращения (11.11.2018)
5. Основные принципы программирования: компилируемые и интерпретируемые языки [Электронный ресурс]. URL: https://tproger.ru/translations/programming-concepts-compilation-vs-interpretation/, дата обращения (11.11.2018)
6. Дорот В. Л., Новикав Ф. А. Интерпретируемый язык программирования, Интерпретация / Толковый словарь современной компьютерной лексики. — 3-е изд. — СПб.: БХВ-Петербург, 2004. — 608 с.
7. Макконнелл С. Совершенный код. — М.: Русская редакция, 2010. — 896 с.
8. Нойбург. М. Программирование для iOS 7. Основы Objective-C, Xcode и Cocoa : Пер. с англ. — М.: ООО «И.Д.Вильямс», 2014 — 384 с.
9. Руководство по разработке под Android [Электронный ресурс]. URL: https://developer.android.com/studio/intro/, дата обращения (11.11.2018)
10. ISO/IEC/IEEE 24765:2010. Словарь системной и программной инженерии [Электронный ресурс]. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:24765:ed-1:v1:en, дата обращения (11.11.2018)
11. Нейропсихологические тесты и программы стимуляции когнитивных функций [Электронный ресурс]. URL: https://www.cognifit.com/ru/science/cognitive-skills/processing-speed, дата обращения (11.11.2018)
12. Савенко И. И. Технология разработки программного обеспечения: конспект лекции — Томск: Изд - во Томского политехнического университета, 2014 . — 67 с.
13. A. Cooper. About Face 3: The Essentials of Interaction Design — 3-е изд. — Indianapolis: Wiley Publishing, 2007. — 651 с.
14. Методы оценки качества пользовательского интерфейса [Электронный ресурс]. URL: http://it-claim.ru/Library/Books/ITS/wwwbook/ist6/ponomarev2/ponomarev2.htm, дата обращения (11.11.2018)
15. A Guide to GOMS Model Usability Evaluation using NGOMSL [Электронный ресурс]. URL: http://www.academia.edu/14542015/A\_Guide\_to\_GOMS\_Model\_Usability\_Evaluation\_using\_NGOMSL, дата обращения (11.11.2018)
16. Ситуация по России: электронные учебники в школах [Электронный ресурс]. URL: https://newtonew.com/school/e-books-at-russian-schools, дата обращения (11.11.2018)