Содержание:

Введение

Интерфейсы служат для осуществления взаимодействия между людьми и окружающим миром. Они помогают разъяснять и освещать, инициировать и демонстрировать взаимоотношения, объединять нас и разлучать, они управляют нашими ожиданиями и предоставляют доступ к функциям. Процесс разработки интерфейсов отнюдь не искусство, ибо они не являются чем-то монументальным. Интерфейсы служат определенной цели, их эффективность может быть точно определена. Однако, они не утилитарны. Лучшие интерфейсы способны вдохновлять, пробуждать, мистифицировать и усиливать наше взаимодействие с миром.

Программное обеспечение должно разрабатываться с учетом требований и пожеланий пользователя — система должна подстраиваться к пользователю. Вот почему принципы проектирования столь важны.

Пользователи компьютера могут иметь удачный опыт, который внушит им уверенность в своих силах и укрепит высокую самооценку при работе с компьютером. Их действия с компьютером могут быть охарактеризованы как "успех порождает успех". Каждый позитивный опыт общения с программой позволяет пользователю расширять область знакомства с программным обеспечением и повышать свой уровень компетентности. Хорошо продуманный интерфейс, подобно хорошему учителю и учебникам, обеспечивает плодотворное взаимодействие пользователя и компьютера. Удачные интерфейсы даже способны помочь человеку выйти из привычного круга программ, которыми он пользуется, и открыть новые, углубить понимание работы интерфейсов и компьютеров.

Принципы разработки интерфейса — это высокоуровневые концепции и представления, которые могут использоваться при проектировании программного обеспечения. Нужно определить, какой из принципов наиболее важен и приемлем для вашей системы.

Приложение разрабатывается для обеспечения работы пользователя, т.е. для того чтобы он с помощью компьютерной программы быстрее и качественнее решал свои производственные задачи.

С точки зрения эргономики, самое важное в программе — создать такой пользовательский интерфейс, который сделает работу эффективной и производительной, а также обеспечит удовлетворенность пользователя от работы с программой.

Целью данной работы является изучение различных вариантов построения интерфейса программ.

Для достижения поставленной цели необходимо решить ряд задач:

- Раскрыть понятие программного интерфейса,
- Изучить процесс эволюции дизайна построения программных интерфейсов,
- Рассмотреть особенности построения интерфейса программ.

Глава 1. Понятие программного интерфейса

Пользовательский интерфейс, интерфейс пользователя – одна из разновидностей интерфейсов, который является совокупностью средств и методов взаимодействия пользователя с вычислительными устройствами (в частности, ПК). Примером реализации пользовательского интерфейса может быть меню на экране телевизора, управление которым осуществляется с помощью пульта дистанционного управления [17, 8 с.].

Интерактивный интерфейс – интерфейс, организованный таким образом, что устройство, получившее команды от пользователя и исполнившее их, выдаёт информацию пользователю информацию с помощью средств, которыми оно располагает (визуально, звуком, тактильно и т.п.). пользователь, в свою очередь, принимает эту информацию и дает устройству следующие команды теми средствами, которыми он располагает (с помощью кнопок, переключателей, регуляторов, сенсора, голоса и т.д.).

Графический пользовательский интерфейс (GUI — Graphical User Interface) - это средства, позволяющие пользователям взаимодействовать с аппаратными составляющими компьютера достаточно комфортным и удобным для себя образом [10]. В течении многих лет для большого количества операционных систем, таких как OS/2, Macintosh, Windows, AmigaOS, Linux, Symbian OS, и т.п., было создано еще большее количество графических интерфейсов.

Интерфейс является совокупностью, т.е. он состоит из элементов, которые также могут состоять из элементов (например, экран дисплея содержит в себе окна, которые содержат панели, кнопки и прочие элементы [17, 22 с.].

Интерфейс характеризуется удобством, эффективностью, понятностью и часто к интерфейсу применяется понятие «дружественный». Дружественный интерфейс предоставляет пользователю наиболее удобный способ взаимодействия с программным обеспечением путем обеспечения логичности и простоты в расположении элементов управления [12, 42 с.]. Принципы дружественного интерфейса:

- обеспечивает право пользователя на ошибку, которое защищают информационно-вычислительные ресурсы системы от непрофессиональных действий на ПК;
- предоставляет широкий набор иерархических меню, систему подсказок и обучения и т.п., которые облегчают процесс взаимодействия пользователя с ПК;
- существование системы «отката», которая позволяет при выполнении действия, результаты которого не удовлетворили пользователя, вернуться к предыдущему состоянию системы.

Составляющие ПИ [23]:

1. Средства:

- вывода информации из устройства – весь спектр доступных воздействий на пользователя (зрительных, слуховых, тактильных, обонятельных и т.п.) – экран, колонки и т.п.;
- ввода информации в устройство – манипуляторы, кнопки, переключатели, датчики и т.п.

При использовании определенных средств ввода интерфейсы разделяются на следующие типы – жестовый, голосовой, брэйн и т.д., а также комбинированные.

1. Методы – набор правил, которые заложены разработчиком устройства, по которым совокупность действий пользователя должна привести к необходимой реакции устройства и выполнения требуемой задачи (так называемый логический интерфейс).

Пользовательский интерфейс включает три основных компонента [8, 127 с.]:

- взаимодействие приложения с пользователем;
- взаимодействие пользователя с приложением;
- язык общения – определяет разработчик программного обеспечения.

Интерфейсы пользователя разделяют на два типа [17, 11-12 с.]:

1. процедурно-ориентированные:

- примитивные интерфейсы;
- интерфейсы меню;
- интерфейсы со свободной навигацией;
- объектно-ориентированные: интерфейсы прямого манипулирования.

Процедурно-ориентированный интерфейс использует традиционную модель взаимодействия с пользователем, которая основана на понятиях «процедура» и «операция» [2, 27 с]. Процедурно-ориентированный интерфейс:

- обеспечивает пользователя функциями, необходимыми для выполнения задач;
- делает акцент на задачи;
- приложения, окна или операции представляются в виде пиктограмм;
- содержание папок и справочников отражается с помощью таблицы-списка.

Примитивный интерфейс организует взаимодействие с пользователем в консольном режиме.

Пользовательский интерфейс (ПИ) часто понимают только как внешний вид программы. Однако на деле пользователь воспринимает через него всю программу в целом, а значит, такое понимание является слишком узким. В действительности ПИ объединяет в себе все элементы и компоненты программы, которые способны оказывать влияние на взаимодействие пользователя с программным обеспечением (ПО) [17, 14 с.]. Это не только экран, который видит пользователь. К этим элементам относятся [8, 98-99 с.]:

- набор задач пользователя, которые он решает при помощи системы;
- используемая системой метафора (например, рабочий стол в MS Windows®);
- элементы управления системой;
- навигация между блоками системы;
- визуальный (и не только) дизайн экранов программы;
- средства отображения информации, отображаемая информация и форматы;
- устройства и технологии ввода данных;

- диалоги, взаимодействие и транзакции между пользователем и компьютером;
- обратная связь с пользователем;
- поддержка принятия решений в конкретной предметной области;
- порядок использования программы и документация на нее.

Интерфейс со свободной навигацией (графический интерфейс) обеспечивает интерактивное взаимодействие с ПО, визуальную обратную связь с пользователем и возможность прямого управления объектом (кнопки, индикаторы, строка состояния). Интерфейс обеспечивает возможность выполнения любых допустимых в конкретном состоянии операций, доступ к которым возможен через различные интерфейсные компоненты («горячие» клавиши и т.д.) [8, 133 с.].

Объектно-ориентированные интерфейсы обеспечивают взаимодействие с пользователем, ориентированное на манипулирование объектами предметной области. Объектно-ориентированный интерфейс [23]:

- обеспечивает пользователя возможностью взаимодействия с объектами;
- делает акцент на входные данные и результаты;
- объекты представляются в виде пиктограмм;
- объекты визуально размещаются в папках и справочниках.

Рисунок 1. Функции объектно-ориентированных интерфейсов [17, 30 с.]

Эффективность работы означает обеспечение точности, функциональной полноты и завершенности при выполнении производственных заданий на рабочем месте пользователя [23]. Создание пользовательского интерфейса должно быть нацелено на показатели эффективности:

- Точность работы определяется тем, в какой степени произведенный пользователем продукт (результат работы), соответствует предъявленным к нему требованиям. Показатель точности включает процент ошибок, которые совершил пользователь: число ошибок набора, варианты ложных путей или ответвлений, число неправильных обращений к данным, запросов и пр.
- Функциональная полнота интерфейса отражает степень использования первичных и обработанных данных, списка необходимых процедур обработки или отчетов, число пропущенных технологических операций или этапов при выполнении поставленной пользователю задачи [8, 132-133 с.]. Этот показатель может определяться через процент применения отдельных функций в РМ [16, 68 с.].
- Завершенность работы над ПИ описывает степень исполнения производственной задачи средним пользователем за определенный срок или период, долю (или длину очереди) неудовлетворенных (необработанных) заявок, процент продукции, находящейся на промежуточной стадии готовности, а также число пользователей, которые выполнили задание в фиксированные сроки [2, 33-34 с.].

Глава 2. Эволюция дизайна построения программных интерфейсов

Большинству интерфейсов популярных ныне операционных систем свойственно интуитивно-понятное графическое оформление с использованием визуальных эффектов, однако так было не всегда [8, 120 с.]. С точки зрения современного пользователя первые GUI были довольно примитивны, хотя, нужно отдать им должное, это не всегда означало отсутствие качественного по тем временам юзабилити.

Традиционно годом рождения GUI принято считать 1973, именно тогда на свет появился первый в полном смысле этого слова персональный компьютер Xerox Alto, в котором использовался графический интерфейс, но было бы несправедливо при этом не упомянуть о его более ранних предшественниках. В 1962 году учёным Айвеном Сазерлендом была создана программа, которую можно считать первым прообразом графических редакторов [21].

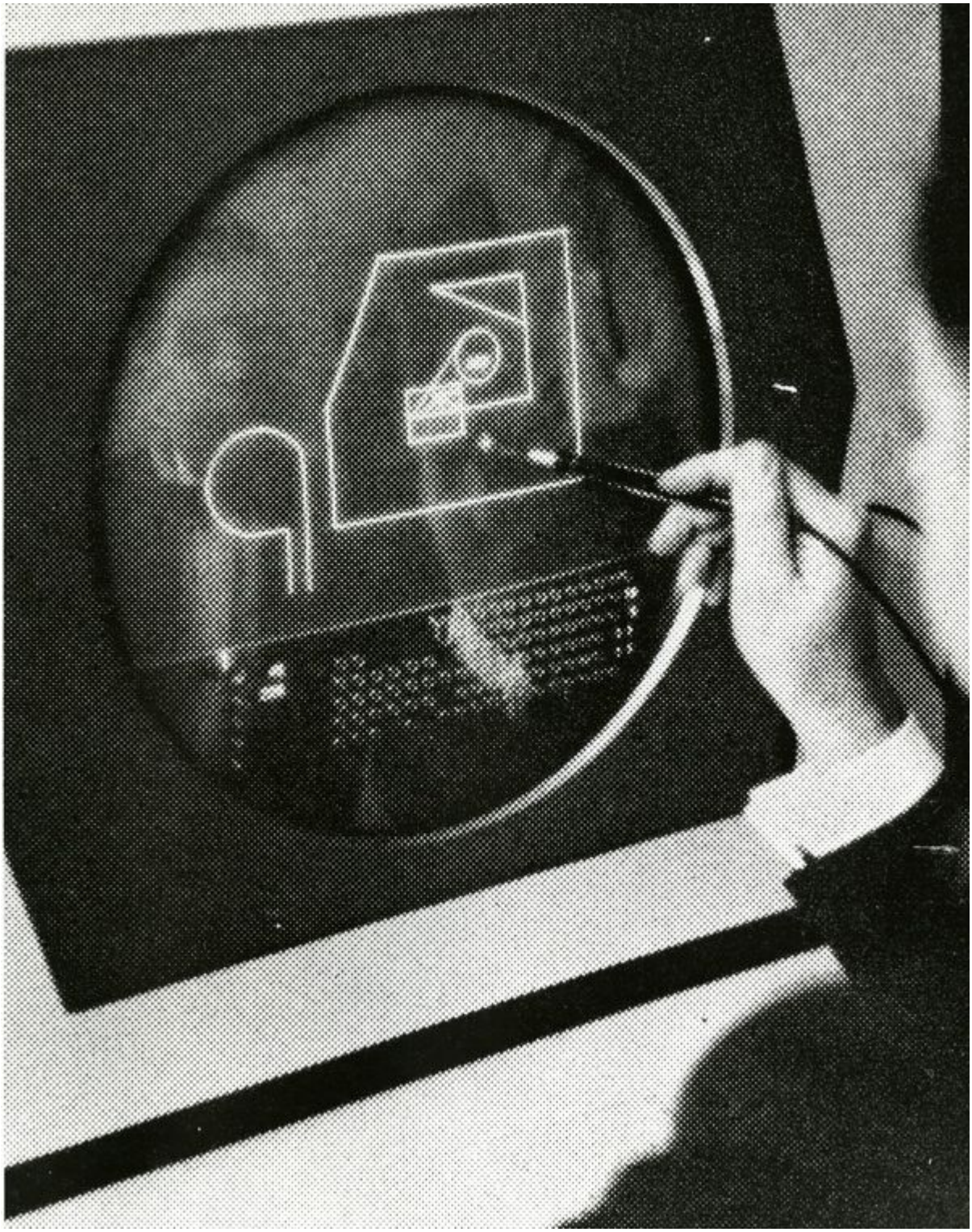

Рисунок 2. Графический интерфейс Sketchpad

Называлась она Sketchpad и позволяла рисовать на экране фигуры световым пером [21]. Спустя шесть лет учёными Стэндфордского института была представлена первая использующая графический интерфейс компьютерная система oN-Line System, в которой уже тогда был заложен концепт современных окон, мышки и

гипертекстовых ссылок [5, 49 с.]. Но oN-Line System была скорее демонстрацией технических возможностей того времени, оставаясь при этом весьма примитивной.

Родоначальником всех ныне существующих графических интерфейсов правильнее считать GUI, разработанный в рамках проекта Xerox Alto — первого персонального компьютера, созданного в 1973 году. Оболочка Xerox Alto была очень проста, но уже тогда в ней присутствовали меню, кнопки и примитивные окна. Был в ней и курсор мыши с присущими ему функциями выделения, копирования и вставки [24].

Рисунок 3. Пользовательская оболочка Xerox Alto

В 1981 году появляется новая система под названием Xerox Star, основанная на той же Xerox Alto, но с более совершенным функционалом и графическим интерфейсом.

Возможно, вы будете удивлены, но рабочий стол Xerox Star мало чем отличался от нынешних десктопов, если, конечно, не брать в расчёт визуальные эффекты. В его основе лежит тот же принцип использования ярлыков для запуска файлов и перехода по каталогам файловой системы [12, 50 с.].

Xerox Star была не единственной на то время операционной системой. В начале 80-х годов свои разработки миру представили компании Apple и Microsoft. Понимая всё значение GUI, но не имея достаточно времени для создания оригинальных оболочек для своих систем, разработчики обеих компаний позаимствовали идеи Xerox Lab, что впоследствии даже привело к конфликту между Стивом Джобсом и Биллом Гейтсом. Джобс обвинил Гейтса в плагиате, что тот, якобы, скопировал интерфейс с Macintosh [14].

Заимствования идей Xerox Lab, однако, вовсе не означают, что никаких попыток создания оригинальных интерфейсов для операционных систем не предпринималось. В 1986 году программистом Джоном Соча был создан Norton Commander - файловый менеджер для MS-DOS, до этого не имевшей практически никакого графического оформления. Роль окон в нем играли панели, делящие экран по вертикали и содержащие списки папок и файлов. В верхней и нижней части менеджера располагались текстовые меню, позволяющие выполнять те или иные операции [8, 123 с.].

Рисунок 4. Интерфейс файлового менеджера Norton Commander [5, 51 с.]

Впрочем, GUI в полном смысле этого слова Norton Commander не являлся. Как и вышедшей в 1988 году его аналог DOS Shell, он относится к псевдографическим интерфейсам, имитирующим графику, оставаясь при этом текстовыми [8, 124 с.]. Тем не менее, оба эти приложения существенно облегчили работу с данными, избавив пользователей от необходимости вводить DOS-команды, чем долгое время и обуславливалась популярность этих программ.

Одновременно с этим, выйдя из команды разработчиков Apple Lisa, в 1982 году Стив Джобс возглавил собственный проект Macintosh. Разработанная для маков система получила название Mac OS [5, 52 с.]. Внешне она была похожа на Apple Lisa, но в ней имелись также и только ей одной присущие особенности, причём касались они как внешнего вида элементов интерфейса, так и самого взаимодействия пользователя с оболочкой. Как и Apple Lisa, MacOS 1.1 была основана на оконном принципе, в ней использовались меню, иконки и диалоги [7, 26 с.].

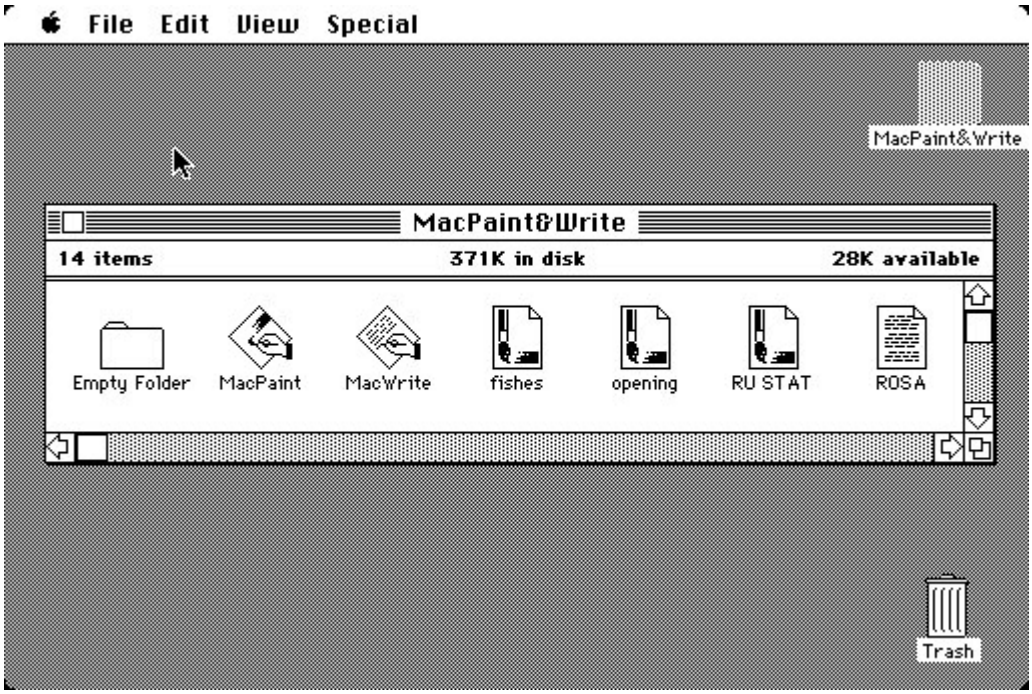

Рисунок 5. Интерфейс MacOS 1.1

Оболочка MacOS 1.1 позволяла быстро переименовывать файлы и папки, выделять их, копировать перетаскиванием в место назначения, одновременно закрывать все окна, хотя закрытие окон не всегда предполагало завершения работы приложения, закрывать программы нужно было правильно — через главное меню системы [7, 28 с.]. При закрытии отредактированных, но не сохранённых файлов появлялось диалоговое окно с запросом на подтверждение сохранения изменений или их

отмены. За 7 лет своего существования Mac OS прошла через множество изменений, но почти все они были незначительными и только в седьмой версии появились нововведения, о которых стоило бы упомянуть [14]. Пожалуй, самое главное из них это поддержка цветов, так как до этого интерфейс системы был практически монохромным. Теперь пользователь мог менять цвет иконок папок и некоторых других элементов, делая их синими, желтыми или красными.

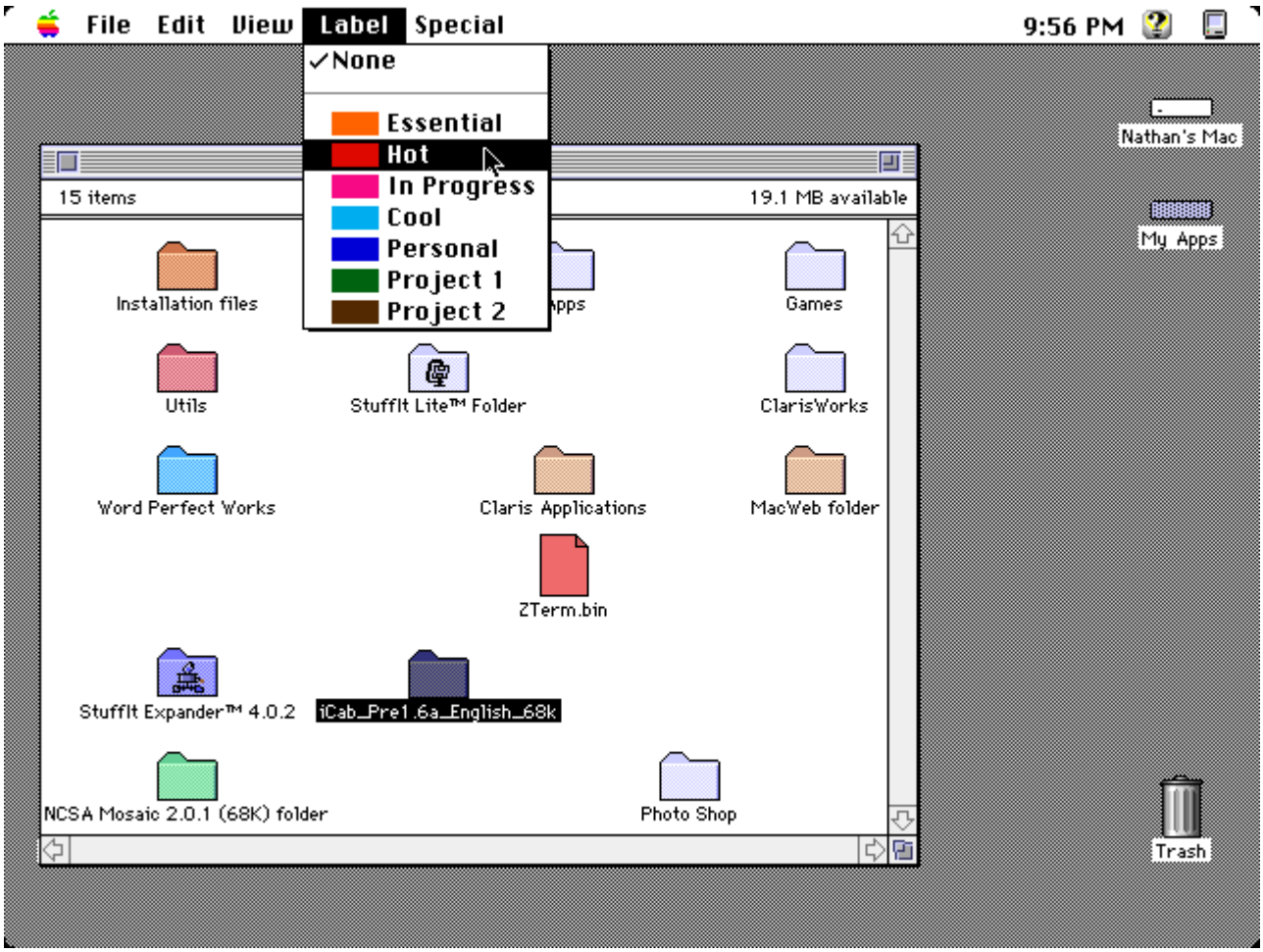

Рисунок 6. Интерфейс Mac OS 7.5.5 [7, 50 с.]

Присутствовали в цветовой гамме Mac OS 7.5.5 и другие оттенки. В это же время становится цветным «яблочный» логотип в левой части главного меню. Из прочих изменений можно отметить показ иконок модулей во время загрузки системы, расширение функционала меню, добавление всплывающих подсказок при наведении на доступные в меню опции, а также реализация доступа к приложениям из единой панели управления [7, 51 с.].

Работа над использованием цвета в графическом интерфейсе была активно продолжена в восьмой версии системы. Системные иконки в Mac OS 8.1 были

цветными по умолчанию, а в самой ОС появилось новое приложение Appearance Manager, позволяющее управлять цветовыми схемами. MacOS 8.1 обзавелась набором фоновых изображений, кроме того, в качестве фонов пользователь мог устанавливать произвольные картинки. В этой же редакции впервые появляется знаменитая платиново-серая тема, ставшая впоследствии визитной карточкой всех последующих версий Mac OS. Другим интересным изменением стало применение к иконкам изометрии, благодаря чему они стали походить на трехмерные объекты, не являясь таковыми на самом деле. Были улучшены настройки отображения содержимого файловой системы — файлы стало можно просматривать в виде списков и значков, размер которых также можно было изменять [7, 33 с.].

Версией 10.14.3, выпущенной в январе 2019 года, завершается история Mac OS на основе оригинальной операционной системы Macintosh и казалось, что в ней должно быть больше нововведений, чем в прошлых версиях. В десятой версии действительно много изменений, но коснулись они по большей части функционала, интерфейс же изменился незначительно [21].

Параллельно Apple, развивалась и индустрия разработки пользовательских интерфейсов компании Microsoft. Еще в 1985 году Microsoft выпустила Windows 1.0 — свою первую операционную систему, основанную на GUI. Система имела 32x32 пиксельные иконки цветную графику [14].

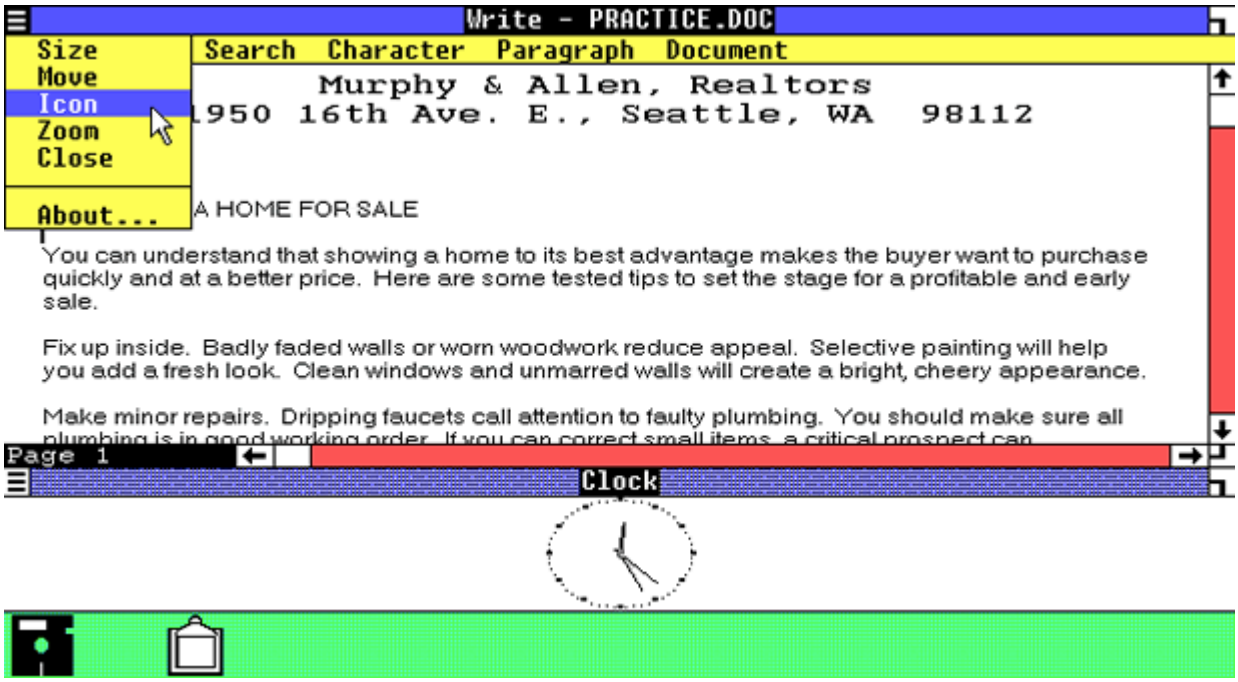

Рисунок 7. Пользовательский интерфейс первой операционной системы Microsoft – Windows 1.0 [22]

В версии Windows 2.0 было значительно улучшено управление окнами. Теперь стало возможным перекрывать, изменять размеры, разворачивать, увеличивать и уменьшать окна.

К версии 3.0 разработчики из Microsoft поняли все реальные преимущества GUI и стали значительно его улучшать. Сама операционная система стала поддерживать стандарты, и расширенный режим для 386 архитектуры, который стал требовать памяти больше чем, 640 килобайт, и больше пространства жесткого диска, в результате стали возможными разрешения, такие как Super VGA 800×600 и XGA 1024×768 [5, 59 с.]. В тоже время, Microsoft пригласили художника и графического дизайнера Сьюзан Каре для разработки дизайна иконок Windows 3.0 и создания уникального образа своего GUI [22].

Версия Windows 3.1 (1992) включала в себя предустановленные TrueType шрифты. На тот момент это фактически определило использование Windows в качестве издательской платформы. Такая функциональность была доступна ранее только в Windows 3.0 с использованием Adobe Type Manager (ATM) — системы работы со шрифтами от компании Adobe. Так же эта версия содержала цветовую схему под названием «Hotdog Stand», содержащую яркие оттенки красного, желтого и черного цветов [22]. Эта схема была создана для облегчения восприятия текстовой и графической информации людьми с нарушениями цветового зрения.

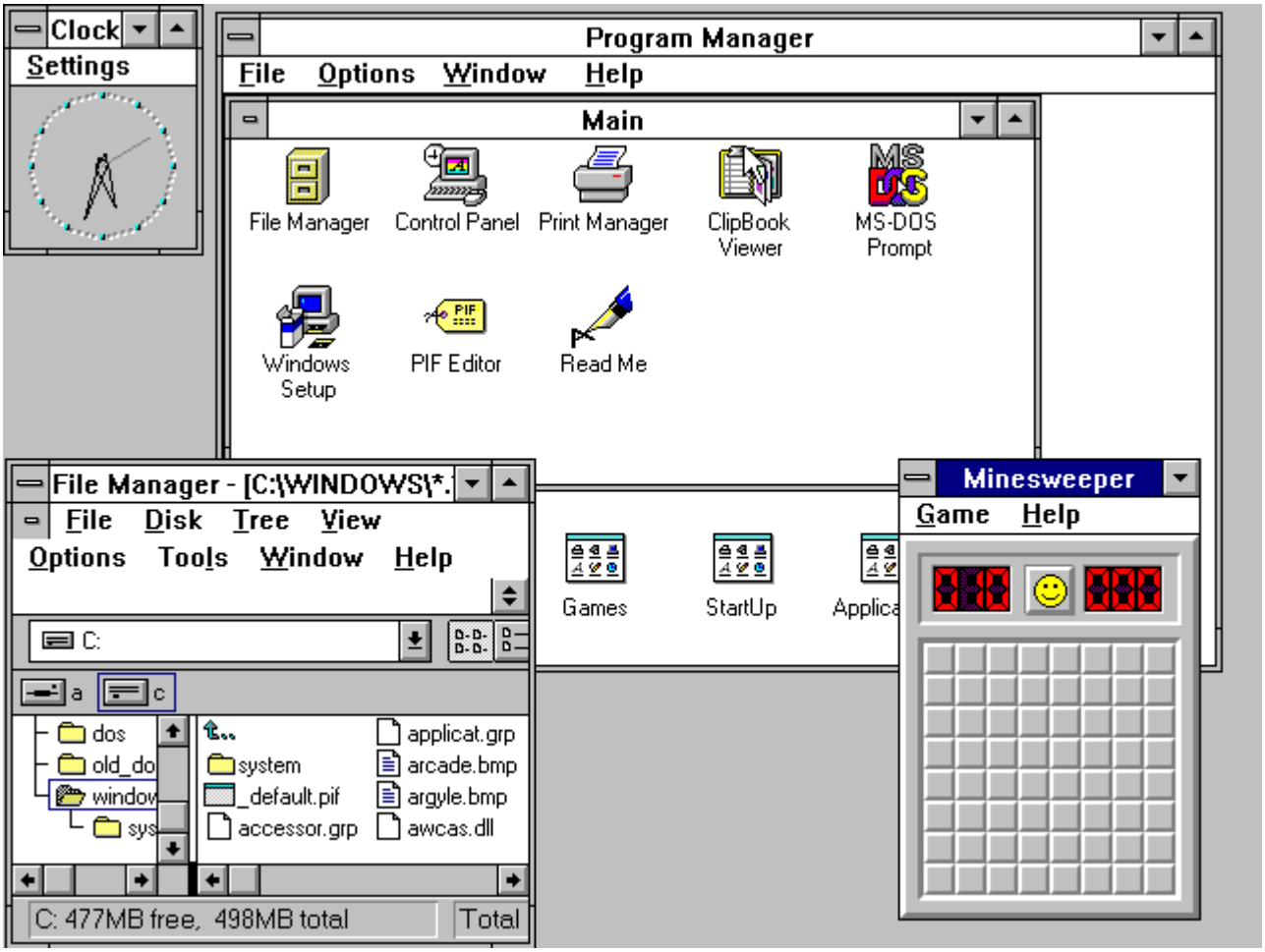

Рисунок 8. Пользовательский интерфейс Windows 3.1.

В Windows 95 был полностью переработан пользовательский интерфейс. Это была первая версия Windows в которой в углу каждого окошка появилась кнопка с крестиком закрывающая его [22]. Были добавлены различные состояния иконок и элементов управления (такие как: доступно, недоступно, выбрано, отмечено и т. д.) [5, 62 с.]. Так же впервые появилась знаменитая кнопка «Пуск». Для Microsoft это был огромный шаг вперед и для операционной системы, и для унификации GUI. В Windows 98 (1998) стиль иконок напоминал Windows 95, но система использовала уже больше 256 цветов для отображения графического интерфейса. Почти полностью изменился Windows Explorer и впервые появился «Active Desktop».

Microsoft старалась полностью изменять пользовательский интерфейс с каждой новой платформой, Windows XP не стал исключением. Стало возможным менять стили для GUI, пользователи могли полностью изменить внешний вид и поведение интерфейса. По умолчанию иконки были размером 48x48 пикселей, и использовали миллионы цветовых оттенков. В Windows Vista появились виджеты и несколько улучшений вместе с отказом от «Active Desktop» [21].

Выпущенная в 2012 году Windows 8 использовал технологию Metro. Metro – это принципиально новый стиль операционной системы, разработанный для новых устройств, где неудобно применять традиционный стиль. Новые технологии (сенсорное управление, планшеты и т.п.) требуют принципиально нового программного обеспечения [14]. В Windows 10 был сделан шаг назад относительно традиционного для Windows меню "Пуск". Если в Windows 8 этого меню совсем не было, а вместо него был «Стартовый экран», то в Windows 10 вернули огрызок меню «Пуск» (каким оно было в Windows 7) и туда же прилепили уменьшенный в размерах «Стартовый экран» [10].

Глава 3. Особенности построения интерфейса программ

Разработка пользовательского интерфейса начинается с вопроса о том, для чего он предназначен, и кто им будет управлять. Хороший дизайнер всегда критически смотрит на окружающую его действительность и делает что-то не просто для процесса, а вдумчиво, по какой-то причине [20]. Правильная разработка интерфейсов - это процесс поиска решений для задач пользователя. Его опыт взаимодействия (UX) влияет на принятие решения о покупке или совершении другого конверсионного действия и может заставить отказаться даже от продукта высокого качества [17, 33 с.]. Интерфейс также решает и задачи бизнеса, потому что от того, насколько им удобно пользоваться клиентам, зависит прибыль компании.

Эргономика включается в процессы разработки и тестирования программного продукта как часть системы качества. Разработка пользовательского интерфейса (ПИ) ведется параллельно дизайну программного продукта в целом и в основном предшествует его имплементации. Процесс разработки ПИ разбивается на этапы жизненного цикла [13, 85-86 с.]:

1. Анализ трудовой деятельности пользователя, объединение бизнес-функций в роли.

2. Построение пользовательской модели данных, привязка объектов к ролям и формирование рабочих мест.

3. Формулировка требований к работе пользователя и выбор показателей оценки пользовательского интерфейса.

4. Разработка обобщенного сценария взаимодействия пользователя с программным модулем (функциональной модели) и его предварительная оценка пользователями и Заказчиком.

5. Корректировка и детализация сценария взаимодействия, выбор и дополнение стандарта (руководства) для построения прототипа.

6. Разработка макетов и прототипов ПИ и их оценка в деловой игре, выбор окончательного варианта.

7. Имплементация ПИ в коде, создание тестовой версии.

8. Разработка средств поддержки пользователя (пользовательские словари, подсказки, сообщения, помощь и пр.) и их встраивание в программный код.

9. Usability тестирование тестовой версии ПИ по набору раннее определенных показателей.

10. Подготовка пользовательской документации и разработка программы обучения.

При разработке пользовательских интерфейсов используются понятия пользовательской истории и пользовательского сценария [16, 67 с.]. Первый термин обозначает способ описания требований к проектируемому продукту в виде нескольких предложений. Второй — детальное описание возможных вариантов поведения пользователя при взаимодействии с интерфейсом. Они нужны для того, чтобы создать правильный продукт [9]. Например, при проектировании формы на сайте дизайнер должен понимать, сколько в ней должно быть полей, что будет достаточным, а что — избыточным. Для этого и нужен пользовательский сценарий. Пример хорошего варианта — несколько строк с подробным описанием ожидаемых действий пользователя и различными реакциями на них элементов интерфейса [10]. Но важно иметь в виду, что записать все пользовательские сценарии до запуска продукта не получится.

В первую очередь стоит понять, что интерфейсы — это язык, который мы уже знаем. Любой пользователь умеет читать и писать на этом языке. С ошибками, конечно, но умеет. Как любой язык, интерфейсы существуют для передачи мыслей [15, 16 с.]. Среди людей выражаться принято связно, поэтому первый шаг — сформулировать идеи и создать сюжет, который захватывает слушателя и дает толчок всему происходящему.

Чтобы понять, что мы имеем в виду под термином «история», давайте представим: Максим становится клиентом автосалона «Звезда Невы» — он купил себе новый автомобиль. При заключении договора в офисе компании сотрудники автосалона сообщают ему о существовании онлайн-системы для клиентов, где он может видеть свои платежи, оплаты по кредиту, напоминания про технический осмотр и страховку, а также новости автосалона. С такого захода мы начинаем «раскручивать» историю. Наиболее важно в процессе придумывания истории проработать все возможные варианты развития событий. Менеджер в салоне выдает Максиму некий ключ доступа к системе. Максим может поделиться этим ключом со своей семьей или друзьями напрямую или опосредованно, разрешая им доступ к системе. При входе в систему Максим видит свой первый платеж в автосалоне, характеристики своего автомобиля, документы на авто, имя своего менеджера. Если его покупка попадает под действие какой-либо промоакции, то он увидит в системе оповещение об условиях акции (например, бесплатная мойка автомобиля класса «люкс» в течение августа). Находясь дома, Максим все больше знакомится с системой. В процессе знакомства система подстраивается под Максима и показывает наиболее актуальную именно для него информацию [9].

Без истории нет ни продукта, ни проекта, ни интерфейса. А связный рассказ позволит объяснить поведение пользователя и понять, чего ему не хватает [13, 70 с.].

Взаимодействуя с компьютером, человек старается не превратиться в машину, а наоборот - очеловечить систему. Какой функционал ни навешивай, без эмоций человеку не будет интересно. Люди ожидают от компьютера новых впечатлений и реакции на свои действия. Хорошие истории вызывают эмоциональный отклик: нравится - не нравится, быстро - медленно, хочу - не хочу.

В процессе проектирования и разработки интерфейсов важно хорошо разбираться в психофизиологии человеческого восприятия. От этих знаний зависит качество будущего продукта [20]. В настоящее время популярность набирает так называемая энергетическая теория, в которой говорится о том, что мозг стремится максимально экономить собственные ресурсы. Он питается углеводами высокой очистки, подготовленные особым образом. Только такие углеводы могут проникать в мозг и питать его. Этот ресурс очень дорогой и ценный, поэтому энергия не должна растрачиваться впустую. Когда есть возможность не активировать какие-то нейроны, мозг старается не делать этого [10]. Поэтому в процессе решения задачи находится наименее энэргозатратное решение. Если мозг удачно с ней справился, выделяется гормон удовлетворения — дофамин. Это важно учитывать

при разработке интерфейсов.

В 20-х годах прошлого столетия в лаборатории Белла ученым-психологом Джорджем Миллером был проведен эксперимент, в процессе которого группы людей решали определенные задачи, используя различное количество объектов. В итоге выяснилось, что чем меньше применяется объектов, тем эффективнее решается задача. Изучив результаты исследования, Миллер вывел правило, что 7±2 объекта — это максимальное количество, которое может вместить кратковременная память человека [3, 71 с.]. Больших чисел мозг начинает избегать для экономии ресурсов. Не так давно появилось новое исследование, где говорится о том, что объектов должно быть не 7±2, а 4±1. Однако существует разница в скорости обработки информации при работе с различными объектами. Более простые обрабатываются быстрее, чем сложные. Задачи с числами решаются быстрее. На втором месте по скорости обработки - цвета, на третьем - буквы, на четвертом - геометрические формы [16, 66 с.]. Многое также зависит от мотивации. Если результат стоит того, чтобы приложить усилия, мозг более охотно решает задачу. При несоблюдении правила 7±2 в процессе разработки интерфейса пользователь теряется в обилии элементов и не знает, какие действия выполнять первыми. Он может отказаться решать слишком сложную задачу и покинуть сайт или приложение.

Важность применения правила 4±1 –заключается в следующем: пользователю приходится решать много задач в повседневной жизни, поэтому интерфейс программы или сайта не должен вызывать у него трудностей. Все нужно выстраивать предсказуемо, логично и просто. При разработке программных интерфейсов необходимо учитывать ресурс человеческого мозга и не заставлять его тратить энергию на ненужные действия [3, 71-72 с.]. Правильная информационная архитектура и таксономия, когда пункты меню сгруппированы понятным образом, помогают пользователю ориентироваться и находить искомое.

Разработчику нужно ставить перед ним задачи, для решения которых достаточно оперировать малым количеством объектов, после чего можно двигаться дальше. Когда пользователь смотрит на страницу, он вычленяет примерно 5 объектов, с которыми впоследствии взаимодействует. Из них он выбирает тот, что быстро приведет его к цели. Работая с объектом, он решает задачу и двигается дальше. В результате его энергия будет сэкономлена, задача решена и пользователь останется доволен, получив приятный опыт взаимодействия с продуктом [19, 189 с.]. Потому применение правила 4±1 делает интерфейс лучше.

У человеческого восприятия есть еще несколько важных особенностей, которые используются при создании интерфейсов. Например, принцип контраста позволяет выделять значимые объекты, делая их более четкими и яркими. Контраст объема заставляет смотреть на более крупный объект [15, 62-63 с.]. Выделенная цветом кнопка большого размера привлекает к себе внимание быстрее, чем маленькая и невзрачная. Противоположным образом оформляются кнопки с нежелательными действиями, например, с отказом от подписки. Для обозначения важного используется размытие фона за ним и воздушная перспектива, что позволяет управлять фокусом пользователя и обращать внимание на конкретный объект [17, 124-125 с.].

Особенности восприятия цветов также используется в разработке интерфейсов программ и приложений. Например, красный для человека означает опасность. Поэтому различные предупреждающие кнопки и знаки, обозначающие действия, которые нельзя отменить, окрашены в этот цвет. Желтый используется для привлечения внимания, зеленый и оранжевый ассоциируются с чем-то безопасным и природным. Но если среди пользователей большой процент дальтоников, использовать цветовые контрасты следует с осторожностью. Один из способов направить взгляд в определенную точку — добавить изображение человеческого лица. Люди с детства приучены распознавать лица и обращать на них внимание, потому всегда реагируют на такую картинку [20].

В процессе чтения активируется несколько обширных зон мозга, ответственных за распознавание, но для восприятия изображения усилий требуется значительно меньше. Поэтому разработчики интерфейсов стараются заменять текст картинками или пиктограммами. Интерфейсы разработки приложений часто сами состоят из иконок и других визуальных элементов. Нужную последовательность считывания информации пользователями можно задавать при помощи правильно подобранных изображений. Но с пиктограммами есть проблема — не каждый человек может расшифровать их значение верно, без процесса обучения [23]. Например, иконка с дискетой, означающая сохранение изменений, до сих пор используется в некоторых программах, но чаще стало применяться изображение облака или облака со стрелкой [10]. Поэтому на первой итерации продукта к новым пиктограммам нужно делать подпись, которая будет разъяснять пользователю, какое действие за ними последует. Затем для пользователей, не сумевших обучиться на первом этапе, в новой версии продукта добавляется подпись, но меньшего размера. В конечном продукте, когда иконка стала привычной, подпись можно убрать. Такие значки экономят место и быстрее распознаются

пользователями, что особенно важно для мобильных приложений и адаптивных сайтов.

Правила контрастности важны не только для графических элементов, но и для текстового контента. Например, в программах для чтения книг есть специальный ночной режим, позволяющий сделать фон черным, а текст — белым. Благодаря этому при вечернем освещении глаза меньше устают от яркого экрана. Тот же принцип используют программисты в процессе написания кода [15, 65 с.]. При цветовом кодировании глаз распознает больше оттенков на темном фоне, особенно красного и фиолетового спектра. Экономить ресурс мозга и быстрее читать текст помогает правильная типографика. Ранее считалось, что человек лучше воспринимает шрифты с засечками, но, согласно новым исследованиям, сейчас читают быстрее привычный шрифт, неважно, с засечками он или без. После разработки концепции, оформления и создания прототипа завершающим этапом проектирования интерфейса является тестирование [19, 202 с.]. После удачного прохождения тестов проект запускается.

Одним из основных понятий в построении пользовательского интерфейса является термин «юзабилити». Если говорить простыми словами, то юзабилити — это то, насколько продукт удобен, понятен и легок в освоении для пользователей. В наше время термин, как правило, употребляется по отношению к системам с графическим интерфейсом пользователя: веб-сайты, софт, операционные системы, приложения для мобильных устройств, интерфейсы банкоматов и терминалов и т.п.

К «железной» составляющей продукта (кнопки, переключатели, форма корпуса и т.п.) обычно применяется термин «эргономичность». Тем не менее, такое разделение не является строгим, и при желании можно говорить о юзабилити чайников, дверных ручек и любых других предметов, с которыми человеку приходится сталкиваться в реальной жизни.

Формальное определение юзабилити по стандарту ISO 9241-11 выглядит следующим образом: «Юзабилити — степень, с которой продукт может быть использован определёнными пользователями при определённом контексте использования для достижения определённых целей с должной эффективностью, продуктивностью и удовлетворённостью» [2].

Само по себе юзабилити не может быть ни хорошим, ни плохим, хотя в разговорной речи этот термин часто употребляется как синоним понятий «хороший», «удобный». Например, фраза «Пользуясь довольно долгое время программой,

пришел к выводу что программе не хватает Юзабилити» подразумевает, что в программе есть некоторые недоработки, из-за чего она не очень удобна [16, 82 с.]. Тем не менее, такое выражение не совсем корректно. Современные разработчики интерфейсов в своих отчетах предпочитают писать о «юзабилити-проблемах», то есть о таких проблемах, которые снижают юзабилити интерфейса, делают его неудобным и непонятым для пользователей.

Далее в определении юзабилити имеются три очень важных понятия:

• определенные пользователи

• определенный контекст использования

• определенные цели

Это значит, что «юзабилити вообще», для всех сразу, не бывает. Вещь, удобная для одного человека, будет совершенно непригодной для другого (вспомните, как вы учили своих пожилых родственников пользоваться мобильным телефоном). Например, в одной крупной компании была разработана серия обучающих видеороликов по работе с внутренней системой документооборота. Инструкции в видеороликах давались при помощи голосового сопровождения, а видеоряд иллюстрировал, что именно надо делать и куда нажимать [6, 109 с.]. Однако разработчики не учли, что ни у одного сотрудника этой компании не было колонок или даже наушников, поэтому обучающие материалы оказались бесполезными. Здесь не был учтен контекст использования, поэтому юзабилити конечного продукта (обучающей программы) оказалось низким.

Эффективность, продуктивность и удовлетворенность — это основные метрики юзабилити, то есть, измерив их, можно определить юзабилити интерфейса [11, 114 с.]. Недостаток этих метрик заключается в том, что их определения недостаточно конкретны — чем, например, эффективность отличается от продуктивности?

При проведении юзабилити-тестирований большинство разработчиков оперирует следующими понятиями:

• эффективность — доля пользователей, выполнивших задачу успешно

• продуктивность — среднее время выполнения задачи (для пользователей, справившихся с задачей)

• удовлетворенность — среднее значение по 5-балльной шкале, выражающее благополучие эмоционального состояния пользователей после выполнения задачи [16, 87-88 с.].

Разработка юзабилити — это методологический подход к созданию сайта или любого другого пользовательского интерфейса. Это практический путь к получению продукта, который работает для пользователя. Разработка юзабилити включает в себя несколько методов, которые последовательно применяются в процессе: сбор требований, разработка и тестирование прототипов, оценка альтернативных вариантов дизайна, анализ проблем пользователей, предложение решений и тестирование сайта (или любого другого интерфейса).

Тестирование юзабилити — это часть процесса разработки юзабилити. В типичном тесте пользователь выполняет некоторое число задач с помощью прототипа (или другой системы), и в это время наблюдатель записывает, что делает пользователь и что он говорит [2, 88 с.]. Обычно тест проводится с одним пользователем или с двумя, работающими вместе. Тестирование может включать сбор таких данных, как последовательность действий пользователя для достижения задачи, ошибки, которые делают пользователи, когда и где у пользователей возникали трудности, насколько быстро пользователи выполняют задачи и насколько им нравиться использовать продукт. Целью большинства тестов является обнаружение любых проблем, которые могут возникать у пользователя, чтобы впоследствии их решить [2, 90-91 с.].

Таким образом, можно утверждать, что юзабилити — это мера качества пользовательского опыта, приобретенного при взаимодействии с продуктом или системой, например, веб-сайтом, программным приложением и т.п.

Общие цели юзабилити говорят о том, что интерфейс должен быть [11, 121-122 с.]:

- Простым в обучении
- Эффективным в использовании
- Легким для запоминания при последующих использованиях
- Удовлетворяющим пользователя.

Все цели юзабилити важны для большинства программных продуктов и сайтов, но можно также выделить и другие для различных ситуаций и аудиторий [6, 110 с.]. Рассмотрим типичный алгоритм достижения высокого уровня юзабилити интерфейса:

- Собираем данные от пользователей. Так как дизайн базируется на потребностях пользователей, надо собрать данные от этих потребностях и о том, насколько существующий веб-сайт (если он есть) удовлетворяет этим потребностям. Существует несколько способов сбора данных, включая формы обратной связи, логи сервера и тестирование юзабилити существующего сайта.
- Разрабатываем прототип. Для пользователя проще взаимодействовать с реальным существующим примером, чем рассуждать, что будет работать лучше всего. Полезные результаты могут быть получены с помощью прототипа сайта, который имеет минимум контента и вообще не имеет графики. Такой простой прототип подходит для первого цикла тестов. Прототип можно использовать для вытягивания у пользователей комментариев, и определения способности прототипа выполнять требуемые задачи [19, 277-278 с.]. Прототипы можно делать на бумаге, в HTML или с помощью специальных средств, таких как MSVisio.
- Собираем, пишем и изменяем контент. Вы должны размещать тот контент, который нужен пользователям сайта. Если у вас есть гора информации, посмотрите, что из нее может быть полезно и понятно пользователям. Чтение с экрана компьютера медленнее, чем чтение с бумаги, многие люди хотят быстро просканировать информацию и прочитать только некоторые маленькие кусочки [4]. Если ваша информация разбита на длинные параграфы, надо разбить ее на маленькие кусочки с подзаголовками. Выкиньте из текста необязательные слова, используйте списки и таблицы, чтобы пользователи могли быстро найти нужную информацию.
- Проводим тестирование. Тестирование юзабилити представляет собой итеративный процесс. Целью тестирования является выяснение того, что помогает пользователям выполнять задачи, а что мешает [6, 110 с.]. Используя прототип как точку отсчета, тестировщики разрабатывают набор сценариев задач, которые пользователи должны будут проделать. После того, как получены результаты тестирования, на их основе можно внести изменения в прототип и провести новые тесты.

Таким образом, к принципам разработки пользовательского интерфейса можно отнести:

- Ясность – это первая и главнейшая цель любого интерфейса. Для того, чтобы эффективно использовать разработанный вами интерфейс, люди должны легко выявлять его, осознавать, зачем они его используют, понимать,

взаимодействие с каким объектом он обеспечивает, представлять, что произойдет при его использовании и, наконец, быть способны взаимодействовать с ним [11, 59 с.]. Хотя при знакомстве с интерфейсом всегда присутствует некая загадка и ожидание, в нем не должно быть места замешательству. Ясность усиливает доверие пользователя и способствует дальнейшему взаимодействию. Лучше сотня понятных экранов, чем один сбивающий с толку.

- Любой ценой удерживайте внимание. Мы живем в мире раздражителей. В наше время сложно погрузиться в чтение – что-то постоянно отвлекает нас и пытается привлечь к себе внимание. А оно бесценно. Не перегружайте боковые панели вашего приложения раздражающими материалами, помните, для чего предназначен экран в первую очередь. Если кто-то читает – дайте ему дочитать до конца, прежде чем показывать рекламу (если без этого нельзя). Уважайте внимание и тогда не только ваши читатели станут счастливее, но и улучшатся ваши показатели [4]. Когда главная цель – побудить к взаимодействию, внимание становится обязательным требованием. Удерживайте его любой ценой.
- Дайте пользователям контроль. Люди наиболее расслаблены, когда уверены, что контролируют себя и собственное окружение. Бездумные программы отнимают у них это ощущение уюта, принуждая к незапланированным взаимодействиям и вводя в замешательство запутанной навигацией и неожиданными результатами [19, 278 с.]. Держите пользователей у руля, регулярно демонстрируя им состояние системы, описывая возможные результаты (если вы сделаете то-то, произойдет то-то) и намекая, чего следует ждать во время следующего шага.
- Прямое взаимодействие является наилучшим. Лучший интерфейс – это его отсутствие, аналогичное нашему миру, где мы способны манипулировать физическими объектами напрямую. Так как это возможно далеко не всегда вследствие растущей информационной насыщенности объектов, мы создаем интерфейсы, дабы облегчить взаимодействие с ними. Очень легко увлечься и добавить слоев больше, чем требуется. Так появляются безумно вычурные кнопки, хромирование, графика, опции, настройки, окна, дополнения и прочая мишура и вот мы уже обнаруживаем себя манипулирующими элементами интерфейса, а не чем-то важным [19, 279 с.]. Откажитесь от этого, стремитесь к изначальной цели прямого управления к созданию интерфейса с наименьшим весом, способным воспринимать наибольшее количество простых человеческих жестов. В идеале интерфейс должен быть настолько легким,

чтобы у пользователя создавалось ощущение прямого взаимодействия с целевыми объектами.

- Для упрощения процесса изучения необходима справка. Буквально — графическая подсказка, объясняющая значение того или иного ЭИ. Полное руководство должно быть частью интерфейса, доступной в любой момент.
- Чаще всего, пользователи в интерфейсе сначала ищут сущность(существительное), а затем действие(глагол) к ней. Следуйте правилу «существительное -> глагол». Например, шрифт -> изменить.
- Одно основное действие на экран. Каждый из проектируемых нами экранов должен быть предназначен лишь для одного действия пользователя. Это делает интерфейс легким в изучении, использовании, а также обслуживании и доработке в случае необходимости. Экран, предназначенный для двух и более основных операций, легко приводит в замешательство [11, 64 с.]. Как статья нуждается в одном ключевом тезисе, также и каждый экран, создаваемый нами, должен служить одному ключевому действию, являющемуся его raison d'etre.
- Делайте следующий шаг естественным. Очень немногие действия должны выполняться в последнюю очередь, так что позаботьтесь о пошаговом переходе пользователя от одного этапа к другому [19, 280 с.]. Старайтесь предвидеть следующее движение пользователя и поддержать его дизайном. Именно это нам и нравится в человеческом общении – открытость для дальнейшего взаимодействия. Не оставляйте пользователей в подвешенном состоянии после того, как они выполнят необходимые вам действия… предоставьте им возможность сделать следующий шаг и продолжить взаимодействие для достижения своих целей.
- Функциональность важнее внешности (действие важнее формы). Люди наиболее комфортно ощущают себя в окружении объектов, поведение которых они способны предсказать – других людей, животных, вещей, программ [19, 281 с.]. Когда кто-то или что-то действует ожидаемым нами образом, мы испытываем к нему расположение. А теперь к делу – внешний вид элементов дизайна должен соответствовать их функциям.

На практике это означает, что любой пользователь должен понимать, как действует тот или иной элемент интерфейса, просто взглянув на него [4]. Если что-то выглядит как кнопка, оно должно действовать как кнопка. Не играйте с базовым взаимодействием приберегите свою фантазию для задач более высокого уровня.

Удачным примером реализации перечисленных принципов можно считать программные продукты фирмы «1С». Возможность самостоятельно менять интерфейс под нужды пользователя существует в продуктах компании «1С». Например, в системе «1С:Предприятия 8.2», используя встроенные средства разработки, администратор может программировать формы, оптимизировать взаимодействие между клиентской и серверной частью и дорабатывать платформу. Прикладные решения доступны не только в локальной сети, но и через интернет, если применять низкоскоростные каналы связи.

Разработка интерфейса в «1С» происходит при помощи встроенного языка, благодаря которому пользователь может динамически перестраивать его части и создавать собственные алгоритмы для обработки данных. Структура определяется набором команд, расположенных в определенной последовательности. В системе нет ограничений по количеству уровней их вложенности [20]. В процессе разработки интерфейса в «1С 8.3» существует механизм настройки программы в зависимости от прав доступа пользователя и его принадлежности к команде. Администратор может настраивать права пользователя и видимость определенных элементов для различных групп, а сам пользователь имеет доступ к дополнительным настройкам при наличии разрешения от администратора [21].

Заключение

Исходя из рассмотренного в работе материала, можно сделать ряд выводов:

1. С точки зрения общих правил, можно отметить, что разработка интерфейса программы обычно начинается с определения задачи или набора задач, для которых продукт предназначен. Простое должно оставаться простым - не нужно усложнять интерфейсы. Постоянно думайте о том, как сделать интерфейс проще и понятнее.

Пользователи не задумываются над тем, как устроена программа. Все, что они видят — это интерфейс. Поэтому, с точки зрения потребителя именно интерфейс является конечным продуктом.

Интерфейс должен быть ориентированным на человека, то есть отвечать нуждам человека и учитывать его слабости. Нужно постоянно думать о том, с какими трудностями может столкнуться пользователь. Думайте о поведении и привычках пользователей. Не меняйте хорошо известные всем ЭИ на неожиданные, а новые

делайте интуитивно понятными. Требования к удобству и комфортности интерфейса возрастают с увеличением сложности работ и ответственности пользователя за конечный результат.

Избегайте двусмысленности. Например, на фонарике есть одна кнопка. По нажатию фонарик включается, нажали еще раз — выключился. Если в фонарике перегорела лампочка, то при нажатии на кнопку не понятно, включаем мы его или нет. Поэтому, вместо одной кнопки выключателя, лучше использовать переключатель(например, checkbox с двумя позициями: «вкл.» и «выкл.»). За исключением случаев, когда состояние задачи, очевидно.

1. Говоря об отдельных элементах ПИ, также стоит отметить ряд особенностей процесса его построения:

- Цвет. Цвета делятся на теплые(желтый, оранжевый, красный), холодные(синий, зеленый), нейтральные(серый). Обычно для ПИ используют теплые цвета. Это как раз связано с психологией восприятия. Стоит отметить, что мнение о цвете — очень субъективно и может меняться даже от настроения пользователя.
- Форма. В большинстве случаев — прямоугольник со скругленными углами. Или круг. Полностью прямоугольные ПИ, лично мне нравятся меньше. Возможно из-за своей «остроты». Опять же, форма как и цвет достаточно субъективна.
- Основные элементы ПИ (часто используемые) должны быть выделены. Например, размером или цветом. Иконки в программе должны быть очевидными. Если нет — подписывайте. Ведь, по сути дела, вместо того чтобы объяснять, пиктограммы зачастую сами требуют для себя объяснений. Старайтесь не делать слишком маленькие элементы — по ним очень трудно попасть.

1. обязательна прозрачная для пользователя навигация и целевая ориентации в программе. Главное, чтобы было понятно, куда идем, и какую операцию программа после этого шага произведет. Ясности и четкости понимания пользователем текстов и значения икон. В программе должны быть те слова и графические образы, которые пользователь знает или обязан знать по характеру его работы или занимаемой должности. Быстроты обучения при работе с программой, для чего необходимо использовать преимущественно стандартные элементы взаимодействия, их традиционное или общепринятое их расположение. Наличия вспомогательных средств поддержки пользователя (поисковых, справочных, нормативных), в том числе и для принятия решения в

неопределенной ситуации (ввод по умолчанию, обход «зависания» процессов и др.).

Для оценки необходимого уровня удобства интерфейса также используются специальные опросники, формуляры, чек-листы, однако к данной работе лучше привлекать специалистов по эргономике.

Список использованной литературы

1. ГОСТ Р ИСО 9241-11-2010. Эргономические требования к проведению офисных работ с использованием видеодисплейных терминалов
2. Баканов А.С. Проектирование пользовательского интерфейса: эргономический подход. Москва: Ин-т психологии РАН, 2009. – 182 с.
3. Баканов А.С. Эргономика пользовательского интерфейса: от проектирования к моделированию человеко-компьютерного взаимодействия. Москва: Ин-т психологии РАН, 2013. - 176 с.
4. Белоусова С.А., Рогозов Ю.И. Анализ подходов к созданию пользовательского интерфейса. // Известия Южного федерального университета. Технические науки, №4, 2014
5. Бережная И.Н. История и философия науки и техники: учебно-методическое пособие для студентов первого курса специальностей. Белгород: Изд-во БГТУ, 2007. - 128 с/
6. Джонсон Д. Умный дизайн: простые приемы разработки пользовательских интерфейсов. Пер. с англ. Е. Шикарева. СПб: Питер, 2012. - 224 с.
7. Иванов С.Р. Стив Джобс. Москва: КоЛибри, 2015. - 95 с.
8. Иванова Г.С. Технология программирования. 3-е изд., стер. Москва: КноРус, 2013. - 333 с.
9. Ким В.Ю. Особенности разработки дизайна пользовательского интерфейса для мобильного приложения. // Новые информационные технологии в автоматизированных системах, №8, 2015
10. Лукьянов Д.В. Разработка графического пользовательского интерфейса. // Новые информационные технологии в автоматизированных системах, №11, 2012
11. Мандел Т. Разработка пользовательского интерфейса. М.: АСТ, 2014. - 412 с.
12. Попов А.А. Технология разработки программного обеспечения: курс лекций. Красноярск: СибГТУ, 2016. - 113 с.
13. Попов А.А. Эргономика пользовательских интерфейсов в информационных системах. Москва: РУСАЙНС, 2016. - 311 с.

14. Сербин В.А. Эволюция пользовательских интерфейсов. // Гуманитарная информатика, № 7, 2013
15. Смирнов А.А. Разработка прикладного программного обеспечения. Москва: МЭСИ, 2012. - 108 с.
16. Сорокин А.А. Объектно-ориентированное программирование: учебное пособие. Ставрополь: Изд-во СКФУ, 2014. - 174 с.
17. Сухоруков К.Ю. Проектирование и разработка пользовательского интерфейса. Саратов: Саратовский гос. технический ун-т, 2016. - 292 с.
18. Терещенко П.В. Интерфейсы информационных систем: учебное пособие. Новосибирск: НГТУ, 2012. - 65 с.
19. Тидвелл Д. Разработка пользовательских интерфейсов. Пер. с англ. Е. Шикарева. - 2-е изд. - Москва: АСТ, 2012. - 478 с.
20. Федечкин Р.С., Французова Ю.В. Разработка интерфейса пользователя для автоматизированной системы. // Известия Тульского государственного университета. Технические науки, №7-2, 2016
21. История развития графического интерфейса. [Электронный ресурс], режим доступа: https://appleinsider.ru/istoriya-apple/istoriya-razvitiya-graficheskogo-interfejsa.html
22. От 1.0 до 10: история развития Microsoft Windows. [Электронный ресурс], режим доступа: https://tproger.ru/translations/microsoft-windows-history/
23. Жакров С. Основы построения интерфейсов. [Электронный ресурс], режим доступа: http://www.interface.ru/home.asp?artId=2634
24. Xerox Alto - CHM Revolution. [Электронный ресурс], режим доступа: https://www.computerhistory.org/revolution/input-output/14/347