Министерство образования и науки Российской Федерации
Вологодский государственный технический университет

Кафедра автоматики и вычислительной техники

ЧЕЛОВЕКО-МАШИННОЕ ВЗАИМОДЕЙСТВИЕ

Рабочая программа и методические указания
к лабораторному практикуму

Факультет электроэнергетический

Специальность: 230105 – программное обеспечение
вычислительной техники и автоматизированных
систем
Направление бакалавриата: 230100 – информатика
и вычислительная техника

Вологда
2011

УДК 621.374.1

Человеко-машинное взаимодействие: рабочая программа и методические указания к лабораторному практикуму.- Вологда: ВоГТУ, 2011 – 27 с.

Приводится рабочая программа дисциплины с указанием тематики и содержания основных разделов, содержание лабораторного практикума и указания по выполнению лабораторных работ, список литературы.

Утверждено редакционно-издательским советом ВоГТУ

Составитель: С. Ю. Ржеуцкая, канд. техн. наук, доцент

Рецензент: А.М. Водовозов, канд. техн. наук, доцент
зав. кафедрой УВС

ВВЕДЕНИЕ

Методические указания предназначены для студентов специальности 230105 — «Программное обеспечение вычислительной техники и автоматизированных систем», но могут быть полезны студентам любых специальностей, которые каким-либо образом связаны с разработкой программных продуктов. Их содержание полностью соответствует стандартной программе курса «Человеко-машинное взаимодействие» ГОС ВПО.

Методические указания содержат четыре главы. В первой из них определяются цели и задачи преподавания дисциплины. Вторая глава содержит рабочую программу лекционного курса с указанием названий и содержания основных тематических разделов. В третьей главе рассматривается содержание лабораторного практикума и порядок выполнения лабораторных работ. Четвертая глава, самая объемная, содержит минимальный объем теоретических сведений и указания по выполнению заданий, составляющих предмет лабораторного практикума.

Следует понимать, что разработка человеко-машинного интерфейса программных продуктов является сложным творческим процессом, поэтому данные методические указания следует воспринимать как общие рекомендации, которые должны быть осмыслены и воплощены в виде пользовательского интерфейса конкретных программных продуктов в процессе их разработки.

1. ЦЕЛЬ И ЗАДАЧИ ДИСЦИПЛИНЫ

1.1. Цель преподавания дисциплины

Основной целью преподавания дисциплины «Человеко-машинное взаимодействие» является развитие профессиональных компетенций студентов в области проектирования, реализации и тестирования человеко-машинных интерфейсов в процессе разработки программной продукции.

В результате изучения дисциплины студенты должны

знать:

- ✓ психологические и физиологические аспекты человеко-машинного взаимодействия,

- ✓ аппаратные средства диалога с пользователем, мультимедиа-устройства,
- ✓ основные принципы и этапы проектирования пользовательского интерфейса,
- ✓ методы тестирования пользовательских интерфейсов,
- ✓ наиболее распространенные инструментальные среды разработки пользовательских интерфейсов и библиотеки визуальных компонентов.

уметь:

- ✓ обобщать и анализировать требования пользователей к организации человеко-машинного интерфейса,
- ✓ проектировать пользовательский интерфейс программных продуктов в соответствии с типовыми сценариями работы пользователей,
- ✓ разрабатывать элементы человеко-машинного интерфейса в процессе реализации программного продукта.
- ✓ выполнять тестирование пользовательского интерфейса

1.2. Взаимосвязь курса с другими дисциплинами

Изучение дисциплины предполагает предварительную подготовку студентов по программированию, полученную в процессе изучения курсов «Программирование и основы алгоритмизации» и «Информатика». Компетенции, сформированные в рамках данного курса, будут использованы в процессе работы над курсовыми и дипломными проектами, а также в будущей профессиональной деятельности в области информационных технологий.

1.3. Требования стандарта к минимуму содержания дисциплины

Ниже приводится выдержка из государственного стандарта по данной дисциплине:

Понятие информационного взаимодействия; психологические аспекты человеко-машинного взаимодействия, уровни сложности и ориентация на пользователя; аппаратные средства графического диалога и мультимедиа-устройства, виртуальные устройства диалога; граф диалога, время ответа и время отображения результата, формальные методы описания диалоговых систем; метафоры пользовательского интерфейса и концептуальные модели взаимодействия; прикладные аспекты человеко-машинного взаимодействия при визуальном проектировании процессов, структур, объектов; инструментальные среды разработки пользовательских интерфейсов.

2. РАБОЧАЯ ПРОГРАММА КУРСА

2.1. Наименование тем лекций и их содержание

Тема 1. Понятие эргатической (человеко-машинной) системы

Предмет инженерной психологии и эргономики. Понятие информационного взаимодействия. Структурная схема человеко-машинного взаимодействия. Физиологический и психологический аспекты. Сущность, факторы, показатели и динамика работоспособности человека-оператора. Методы измерения работоспособности по частным показателям. Требования к эргатической системе с точки зрения психологии и эргономики.

Тема 2. Аппаратные средства человеко-машинного взаимодействия

Устройства отображения информации: основные характеристики с точки зрения человеко-машинного взаимодействия. Устройства «виртуальной реальности». Устройства ввода: клавиатуры, сканеры, графические планшеты (дигитайзеры), MIDI-клавиатуры, Web-камеры. Устройства управления: мышь, игровые манипуляторы (джойстики). Современное состояние и перспективы развития аппаратных средств человеко-машинного взаимодействия.

Тема 3. Концепции пользовательского интерфейса (ПИ)

Понятие и основные составляющие пользовательского интерфейса. Основные концепции пользовательского интерфейса: технологическая и метафористическая. Достоинства и недостатки использования метафор в пользовательском интерфейсе. Примеры удачных и неудачных метафор.

Тема 4. Классификация ПИ

Интерфейс командной строки (CLI): достоинства и недостатки. Графический интерфейс (GUI). Двумерный графический интерфейс (WIMP-интерфейс). Основные элементы двумерного графического интерфейса: окна, меню, иконки, списки, курсоры и т.д. Трехмерный графический интерфейс: перспективы развития.

Тема 5. Основы проектирования ПИ

Требования к ПИ: максимально возможная скорость работы пользователя, минимум ошибок пользователя, легкость обучения, психологиче-

ский комфорт. Типовые приемы проектирования интерфейса с целью выполнения данных требований. Этапы проектирования ПИ, основные задачи, выполняемые на каждом этапе.

Тема 6. Библиотеки визуальных компонентов и среды разработки ПИ

Эволюция библиотек визуальных компонент: библиотеки Turbo Vision и MacApp. Библиотека Microsoft Foundation Classes (MFC), Visual Component Library (VCL — Borland), пакеты Java, средства для разработки Web-интерфейса. Обзор современных сред визуальной разработки приложений.

Тема 7. Проблемы и тенденции развития человеко-машинного интерфейса

Мультимедиа среды и мультисенсорные системы: речевой интерфейс, звуковые сигналы, распознавание текстов, анимация и видеофрагменты, распознавание жестов, компьютерное зрение.

Системы виртуальной реальности: язык виртуальной реальности (VRML), функции браузеров и поведение в виртуальной среде, виртуальные многопользовательские среды.

3. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Цель лабораторного практикума – формирование навыков проектирования и реализации пользовательских интерфейсов в процессе разработки программных продуктов.

Лабораторный практикум включает 8 двухчасовых лабораторных работ, объединенных общей тематикой и предусматривающих решение следующих задач:

1. Спроектировать и реализовать в среде визуального программирования пользовательский интерфейс автоматизированных рабочих мест (АРМ) информационной системы для конкретной предметной области (по вариантам). Среда разработки — Delphi, C++ Builder, Visual C++ (на усмотрение студента по согласованию с преподавателем).
2. Проанализировать типовые сценарии работы пользователей для каждого АРМ в рамках своего индивидуального задания и показать, как они реализованы в интерфейсе.
3. Рассчитать время выполнения пользователем типовых сценариев работы для каждого рабочего места. Выполнить экспериментальную проверку рассчитанного времени на нескольких испытуемых.

4. Проанализировать все возможные ошибки пользователя при вводе данных и предложить способы сокращения количества человеческих ошибок при работе с разработанной системой.
5. Разработать инструкцию для пользователей реализованных автоматизированных рабочих мест. Проанализировать сложность освоения интерфейса АРМ пользователем и экспериментально проверить на нескольких испытуемых.

Варианты заданий на разработку пользовательских интерфейсов совпадают с заданиями на курсовые проекты по дисциплине «Базы данных». В целях организации междисциплинарных связей отчеты по лабораторным работам по ЧМВ сдаются вместе с курсовым проектом по дисциплине «Базы данных», но защищаются отдельно.

4. КРАТКИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

В данном разделе будут представлены краткие теоретические сведения и указания по проектированию, реализации и тестированию пользовательских интерейсов в процессе разработки программных продуктов, которыми следует пользоваться в процессе выполнения лабораторных работ по индивидуальным заданиям.

4.1. Общие сведения

Под *пользовательским интерфейсом* (ПИ) программы будем понимать совокупность элементов, позволяющих пользователю программы управлять ее работой и получать требуемые результаты.

ПИ часто понимают только как внешний вид программы. В действительности он включает в себя все аспекты, которые оказывают влияние на взаимодействие пользователя и системы. ПИ состоит из таких составляющих:

- набор задач пользователя, которые он решает при помощи системы
- используемые системой метафоры (например, рабочий стол, корзина в MS Windows и т.п.)
- элементы управления системой
- навигация между блоками системы
- дизайн экранов программы.

4.2. Классификация пользовательских интерфейсов

Классы интерфейсов тесно связаны с используемыми базовыми техническими средствами человеко-машинного взаимодействия (таблица 4.1).

Таблица 4.1

Классификация ПИ

Классы интерфейса	Подклассы	Управляющие средства
Символьный	Командный интерфейс	«Вопрос-ответ»
		Командная строка
Графический	Двухмерный	Экранные формы
		Управляющие клавиши
		Меню
		Графические элементы управления
		Прямое манипулирование
	Трехмерный	Конические деревья

Сегодня развиваются такие новые классы интерфейсов, как SILK (речевой), биометрический (мимический) и семантический (общественный).

Начал получать распространение и новый вид пользовательского интерфейса – тактильный**.** Пока эта область еще не достаточно изучена, основанная на тактильных ощущениях аппаратура появилась совсем недавно.

В настоящее время оформилось два принципиально различных подхода к организации пользовательского интерфейса. Первый, исторически более ранний подход состоит в предоставлении пользователю командного языка. Этот подход известен как *интерфейс командной строки* (Command Line Interface - CLI).

Альтернативный подход состоит в символическом изображении доступных действий в виде картинок – икон (icons) на экране и предоставлении пользователю возможности выбирать действия при помощи мыши или другого координатного устройства ввода. Этот подход известен как *графический пользовательский интерфейс* (Graphical User Interface - GUI). Один из подклассов GUI (двухмерный) принято обозначать аббревиатурой WIMP (Windows-Icons-Menus-Pointing device).

Разработчики современных ОС обычно предоставляют средства для реализации обоих подходов и, зачастую, оболочки, использующие оба типа интерфейсов. Сравним их достоинства и недостатки.

Аргументы в пользу CLI

Удачно спроектированные командные языки обеспечивают: высокую скорость обработки, эффективность и экономию использования ресурсов системы. Важными преимуществом хороших командных языков по сравнению с GUI является их алгоритмическая полнота: в GUI пользователь ограничен теми возможностями, для которых разработчик программы нарисовал иконки или сочинил пункты в меню. Командные же языки могут использоваться для решения любых алгоритмизуемых задач, в том числе и таких, о которых разработчики языка никогда и не задумывались.

Любопытно, что последнее достижение в области ПИ - распознавание речи и голосовое управление - означает, по существу, возврат к командному языку, с той лишь разницей, что команды произносятся.

Аргументы в пользу GUI

Командные языки требуют затрат времени и усилий для изучения. Напротив, графический интерфейс предоставляет новому пользователю возможность быстро окинуть взглядом доступные возможности и выбрать желаемую. Во многих случаях наглядность вариантов оказывается важнее богатства возможностей.

Хорошо продуманные интерфейсы обеспечивают *почти* такую же гибкость в комбинации операций, как и командные языки. Возможность же записывать и вновь проигрывать последовательности действий (например, макросы) во многих ситуациях может отлично заменить командные файлы.

Относительно высокие накладные расходы, связанные с графическими интерфейсами, не являются таким уж большим злом. Современные цены на аппаратуру достаточно низки для того, чтобы большая эффективность изучения и использования графической системы быстро себя окупала.

Недостатки WIMP-интерфейсов

Во-первых, чем более сложным является приложение, тем труднее осваивать интерфейс, причем эти трудности возрастают нелинейно. Многие современные настольные приложения столь объемны, что пользователь начинает даже отказываться от новейших версий, продолжая использовать то малое подмножество возможностей, которое удалось изучить.

Во-вторых, пользователи проводят слишком много времени, манипулируя интерфейсом, а не работая с самим приложением. Квалифицированные пользователи часто бывают раздражены слишком большим количеством слоев, возникающих в процессе point and click

(использование сокращенных комбинаций клавиш - это суррогатный метод решения этой проблемы).

В-третьих, недостатком WIMP-интерфейсов является то, что они никак не используют такие каналы взаимодействия, как речь, слух и прикосновения.

*Х*орошая программная система должна предоставлять *оба* интерфейса. В рамках лабораторного практикума мы ограничимся только WIMP-интерфейсом как наиболее распространенным в современных программных продуктах.

4.3. Рекомендации по организации WIMP-интерфейса

В процессе проектирования и реализации ПИ следует руководствоваться перечисленными ниже принципами.

1. Естественность (интуитивность)

Работа с системой не должна вызывать у пользователя сложностей в поиске необходимых элементов интерфейса для управления процессом решения поставленной задачи.

2. Непротиворечивость

Если в процессе работы с системой пользователем были использованы некоторые приемы работы с некоторой частью системы, то в другой части системы приемы работы должны быть идентичны. Работа с системой через интерфейс должна соответствовать установленным, привычным нормам (например, использование клавиши Enter).

3. Неизбыточность

Это означает, что пользователь должен вводить только минимальную информацию для работы или управления системой. Например, пользователь не должен вводить незначимые цифры (00010 вместо 10). Аналогично, нельзя требовать от пользователя ввести информацию, которая была предварительно введена или которая может быть автоматически получена. Желательно использовать значения по умолчанию, где только возможно, чтобы минимизировать процесс ввода информации.

4. Непосредственный доступ к системе помощи

В процессе работы необходимо, чтобы система обеспечивала пользователя необходимыми инструкциями. Система помощи отвечает трем основным аспектам - качество и количество обеспечиваемых команд; характер сообщений об ошибках и подтверждения того, что система делает.

5. Гибкость

Интерфейс системы должен соответствовать уровню подготовки различных пользователей. Для неопытных пользователей интерфейс

может быть организован как иерархическая структура меню, а для опытных пользователей как команды, комбинации нажатий клавиши.

Размещение информации на экране

Количество информации, отображаемой на экране, называется экранной плотностью. Исследования показали, что, чем меньше экранная плотность, тем отображаемая информация более доступна и понятна для пользователя. Однако, опытные пользователи могут предпочитать интерфейсы с большой экранной плотностью. Информация на экране может быть сгруппирована и упорядочена в значимые части. Это может быть достигнуто с использованием фреймов, методов типа цветового кодирования, рамок, негативного изображения или других методов для привлечения внимания.

Выделение элементов интерфейса

Для привлечения внимания к каким-либо элементам интерфейса можно воспользоваться выделением этих элементов на фоне других. Однако не стоит переусердствовать с этим методом, поскольку большое количество ярких элементов может вызвать дискомфорт у пользователя. Существует несколько способов выделения элементов:

1. движение (мигание или изменение позиции). Очень эффективный метод, поскольку глаз имеет специальный детектор для движущихся элементов;
2. яркость. Не очень эффективный метод, так как люди могут обнаружить всего лишь несколько уровней яркости;
3. цвет - использование цвета может быть чрезвычайно эффективно;
4. форма (символ, шрифт, форма символа). Используется для того, чтобы отличить различные категории данных;
5. использование различных алфавитов (шрифтов) в различных формах;
6. размер (текста, символов). Обычно применяют увеличение выделенного объекта в 1.5 раза;
7. оттенение (различная текстура объектов). Эффективный метод для привлечения внимания к какой-либо части экрана;
8. окружение (подчеркивание, рамки, инвертированное изображение). Очень эффективный метод, если использовать его в разумных количествах.

Использование цвета при проектировании эргономичного интерфейса

Цвет может улучшить интерфейс пользователя, но для многих систем использование цвета практически не влияет на эффективность работы пользователя. Основное назначение цвета - в создании

интерфейсов, более интересных для пользователей. Однако, имеются случаи, где цвет может помочь проектировщику интерфейса пользователя. Это наиболее эффективно, когда цвет используется для:

1. группировки информации;
2. выделения различий между информацией;
3. выделения простых сообщений (ошибки, состояния и т.д.)

Цвет – мощный визуальный инструмент, его необходимо использовать очень острожно, чтобы не вызвать дискомфорта у пользователя ошибочными цветовыми комбинациями. Приведем некоторые принципы использования цвета, которыми нужно руководствоваться при проектировании эргономичного интерфейса:

1. необходимо ограничить число цветов до 4 на экране и до 7 для последовательности экранов; для неактивных элементов нужно использовать бледные цвета;
2. если цвет используется для кодировки информации, необходимо удостовериться, что пользователь правильно понимает код, например, просроченные счета выделяются красным цветом, а непросроченные – зеленым;
3. необходимо использовать цвета согласно представлениям пользователя, например, для картографа зеленый - лес, желтый - пустыня, синий - вода. Для химика, красный – горячий, синий – холодный;
4. для отображения состояния: красный = опасность/стоп, зеленый = нормально/продолжение работы, желтый = предостережение;
5. для привлечения внимания наиболее эффективны белый, желтый и красный цвета;
6. для упорядочения данных можно использовать спектр 7 цветов (радуга);
7. для разделения данных необходимо выбрать цвета из различных частей спектра (красный / зеленый, синий / желтый, любой цвет / белый);
8. для группировки данных, объединения и подобия нужно использовать цвета, которые являются соседями в спектре (оранжевые / желтые, синие / фиолетовые).

Непротиворечивость и стандартизация

Данные на экране следует располагать таким образом, чтобы пользователь знал, где найти и где ожидать вывода необходимой информации. Информация, на которую следует немедленно обратить внимание, должна всегда отображаться в видном месте, чтобы захватить внимание пользователя (например, предупреждающие сообщения

и сообщения об ошибках). Информация, которая необходима не очень часто (например, средства справки) не должна отображаться, но должна быть доступна, когда потребуется. Например, иконка Справки или соответствующая опция меню должна быть доступна на каждом экране.

Тексты и диалоги

Приведем принципы, которыми нужно руководствоваться при создании текстовых диалогов и отображений:

1. текст в нижнем регистре читается приблизительно на 13% быстрее чем текст, который напечатан полностью в верхнем регистре;
2. символы верхнего регистра наиболее эффективны для информации, которая должна привлечь внимание.
3. выровненный по правому краю текст труднее читать, чем равномерно распределенный текст с невыровненным правым полем;
4. оптимальный интервал между строками равен или немного больше, чем высота символов.

Меню

Меню – набор опций, отображаемых на экране, где пользователи могут выбирать и выполнять действия, тем самым производя изменения в состоянии интерфейса. Достоинство меню в том, что пользователи не должны помнить название элемента или действия, которое они хотят выполнить – они должны только распознать его среди пунктов меню. Таким образом меню может использовать даже неопытный пользователь. Однако, проект меню должен быть тщательно продуман – чтобы меню было эффективным, названия пунктов меню должны быть очевидными.

В процессе проектирования системы меню приложения необходимо принять наилучший способ отображения меню, чтобы оно было понятно и легко в использовании. Обычно команды меню упорядочены некоторым иерархическим способом. Основная проблема состоит в том, чтобы правильно распределить различные пункты меню по различным уровням и правильно их сгруппировать.

Исследования показывают, что имеются четыре варианта для организации меню:

1) алфавитный;
2) категорийный;
3) в соответствии с нормальными соглашениями;
4) в соответствии с частотой использования.

Основные принципы создания меню

1. Структура меню должна соответствовать структуре решаемой системой задачи, организация меню должна отразить наиболее эффективную последовательность шагов, чтобы достичь решения поставленной задачи;
2. Пункты Меню должны быть краткими, грамматически правильными и соответствовать своему заголовку в меню.
3. Выбор пунктов меню должен быть обеспечен несколькими способами – с помощью клавиатуры, с помощью мыши, а также через другие объекты пользовательского интерфейса. Необходимо использовать легко запоминаемые сочетания клавиш для более быстрого доступа к пунктам меню, поскольку это очень экономит время.

Формы

Формы – основной элемент интерфейса. Основные принципы проектирования форм:

1. Форма проектируется для удобного, понятного и скорейшего достижения решения поставленной задачи. Если форма переносится из бумажной формы, то передвижение по смежным полям не должно вызывать затруднений у пользователя.
2. Размещение информационных единиц на пространстве формы должно соответствовать логике ее будущего использования: это зависит от необходимой последовательности доступа к информационным единицам, частотой их использования, а также от относительной важности элементов.
3. Важно использовать незаполненное пространство, чтобы создать равновесие и симметрию среди информационных элементов формы, для фиксации внимания пользователя в нужном направлении.
4. Логические группы элементов необходимо отделять пробелами, строками, цветовыми или другими визуальными средствами.
5. Взаимозависимые или связанные элементы должны отображаться в одной форме.

Дизайн заголовков и полей

1. Для отдельных полей заголовок должен быть выровнен по левому краю; для полей списков, заголовок должен быть выше и левее по отношению к основному полю, числовые поля выравниваются по правому полю.

2. Длинные колоночные поля или длинные столбцы с одиночными полями необходимо объединять в группы по пять элементов, разделяемых пустой строкой - это помогает пользователю мысленно обрабатывать информацию по выделенным группам.
3. В формах с большим количеством информации необходимо использовать названия разделов, которые однозначно свидетельствуют о характере принадлежащей им информации.
4. Необходимо четко разделить отображение заголовков и полей ввода, поскольку такая путаница может вызвать дискомфорт у пользователя.
5. Поля, необязательные для заполнения, либо не имеющие особой важности, должны отличаться визуально (цветом или другими эффектами) от полей важных и обязательных для заполнения.

Форматы ввода

1. Необходимо обеспечить ввод значений по умолчанию во все поля, которые это допускают.
2. Можно назначить клавиши или коды для ввода часто повторяющихся значений.
3. Не объединяйте поля ввода чисел и символов, поскольку числовые и алфавитные клавиши находятся неудобно относительно друг друга на клавиатуре.
4. Поля ввода должны быть короткими, насколько это возможно.
5. Необходимо исключить частое переключение между верхним и нижним регистрами для ускорения ввода данных.
6. Нельзя требовать от пользователя ввода незначимых цифр (например, вместо 00000010 пользователь должен ввести только 10).

Организация системы навигации и системы отображения состояний

Навигация обеспечивает возможность перемещения между различными экранами, информационными единицами и подпрограммами. В полноценной системе пользователь всегда может получить информацию о состоянии системы, процесса выполнения или активной подпрограмме.

Существует ряд навигационных средств и приемов, которые помогают пользователю ориентироваться в системе. Они включают: использование заголовков страниц для каждого экрана; использование номеров страниц; номеров строк и столбцов; отображение текущего имени файла вверху экрана.

В интерфейсах с меню можно использовать иерархически-структурированное меню. Для выхода из подменю нужно применять несложные действия. Диалоговые интерфейсы сами по себе защищают пользователя от ошибочных действий. Информация состояния программы обычно отображается внизу экрана и содержит в себе данные количеств записей, числе обработанных единиц, процессе печати, очереди печати и т.д.

Проектирование сообщений

Сообщения необходимы для направления пользователя в нужную сторону, подсказок и предупреждений для выполнения необходимых действий на пути решения задачи. Они также включают подтверждения действий со стороны пользователя и подтверждения, что задачи были выполнены системой успешно либо по каким-то причинам не выполнены. Сообщения могут быть обеспечены в форме диалога, экранных заставок и т.п.

Сообщения могут быть помещены в модальные диалоговые окна, которые вынуждают пользователя ответить на вопрос прежде, чем может быть предпринято любое другое действие, потому что все другие средства управления заморожены. Это может быть полезно, когда система должна вынудить пользователя принять решение перед продолжением работы. Немодальные диалоговые окна позволяют работать с другими элементами интерфейса, в то время как само окно может игнорироваться.

Предотвращение, обнаружение и исправление ошибок

Ошибки могут быть классифицированы как:
1) ошибки, которые основаны на неправильном понимании действия или порядка действий;
2) ошибки, которые возникли случайно, непреднамеренно, например опечатка при вводе текста;
Ошибки второго вида могут быть разделены на шесть подвидов:

- ошибки неточности выбора опции (например, пользователь случайно нажал кнопку "Выход" и программа закрылась);
- ошибки управления данными (например, присвоение ошибочного имени файла из-за неточности отображения последнего);
- ошибки ассоциативного характера (например, сохранение файла с именем какого-либо человека, так как пользователь думал о нем в момент сохранения);
- ошибка потери активности, когда пользователь забывает необходимую последовательность действий;

- ошибка режима или состояния - когда пользователь думает, что он находится в одном состоянии, но - фактически в другом, например, режим вставки взамен режима печати поверх текста в текстовом процессоре.

Пользователь всегда будет делать ошибки, даже в отличной программной системе, поэтому в разрабатываемой системе всегда должна быть предусмотрена защита от ошибок. Техника защиты от ошибок включает в себя следующие аспекты.

1. Принудительные действия в системе, которые предотвращают или затрудняют появление ошибок;
2. Обеспечение хороших и информативных сообщений об ошибках;
3. Использование обратимых действий, которые позволяют пользователям исправлять их собственные ошибки;
4. Обеспечение нормальной диагностики системы, в процессе которой пользователю объясняется, в чем суть ошибки и пути ее исправления.

Обработка ошибок в формах ввода

Следует руководствовться следующими основными принципами.

1. Обеспечить возможность посимвольного редактирования введенных записей для исправления ошибок ввода (опечаток);
2. Если ошибка была обнаружена системой, желательно вернуть курсор в поле с ошибочными данными и каким-либо образом выделить это поле визуально;
3. Обеспечить значимые сообщения об ошибках, использующие стиль языка пользователя и соответствующую терминологию;
4. Обеспечить сообщения об ошибках, которые объясняют их и предлагают пути устранения.

4.4. Этапы разработки ПИ

Проектирование ПИ неразрывно связано с разработкой других составляющих программного продукта, поэтому разработка элементов интерфейса выполняется на всех этапах жизненного цикла программы.

4.4.1. Постановка задачи (этап анализа)

На этой стадии собираются и анализируются данные о пользователях, формализуется функциональность и определяются объективные критерии успеха проекта.

Общая характеристика пользователей

Описываются следующие свойства аудитории системы:

- Характеристики пользователей: их опыт работы с компьютером, знание предметной области, мотивы, размер/важность групп пользователей, образцы (типовые ситуации) использования;
- Цели и задачи пользователей;
- Технология разработки и платформа, на которой будут работать пользователи;
- Среда, в которой будет создаваться и использоваться проект (физическая, рыночная, организационная и культурная).

Формализация бизнес-ролей пользователей

Выделяются основные роли пользователей с относящимися к этим ролям функциям. Проводятся собеседования с каждым из представителей определенной роли на предмет выявления особенностей данной роли и выяснения, какие дополнительные возможности следует предусмотреть.

На этом этапе можно применять метод наблюдения за людьми, выполняющими свою задачу, пользуясь уже имеющимися инструментами.

Формализация функциональности

Основываясь на информации, выработанной на предыдущих этапах, окончательно формируется список функциональных возможностей новой версии системы и окончательно формулируется ТЗ.

Формализация сценариев действий пользователей

На этом этапе частично изучаются, частично разрабатываются типовые сценарии действий пользователя. Цель этого этапа – написать словесное описание взаимодействия пользователя с системой, не конкретизируя, как именно проходит взаимодействие, но уделяя возможно большее внимание всем целям пользователей. Количество сценариев может быть произвольным, главное, что они должны включать все типы задач и быть реалистичными.

Обзор интерфейса конкурирующих систем

Большая часть аудитории любой системы обладает навыками использования нескольких конкурирующих систем; если разрабатываемый интерфейс полностью несхож с конкурентами, пользователям придется переучиваться. Кроме того, конкурирующие системы часто

содержат эффективные решения, которые полезно перенять (или, что чаще случается, учесть при проектировании интерфейса).

Формализация привычек и ожиданий пользователей

На данном этапе изучаются субъективные ожидания пользователей от системы. Без этого исследования трудно или невозможно предугадать отношение пользователей к будущей системе.

4.4.2. Этап проектирования

Проектирование структуры экранов системы

На этом этапе, основываясь на сценариях работы и ролях пользователей, формируется структура экранов системы, т.е. определяется количество экранов, функциональность каждого из них, навигационные связи между ними, формируется структура меню и других навигационных элементов.

Существует три основных вида связи между блоками. Это логическая связь, связь по представлению пользователей и процессуальная связь.

Логическая связь. Логическая связь определяет взаимодействие между фрагментами системы с точки зрения разработчика. Полученные связи очень существенно влияют на навигацию в пределах системы (особенно, когда система многооконная). Поэтому чтобы не перегружать интерфейс, стоит избегать как слишком уж отдельных блоков (их трудно найти), так и блоков, связанных с большим количеством других.

Связь по представлению пользователей. В информационных системах, когда необходимо гарантировать, что пользователь найдет всю нужную ему информацию, необходимо устанавливать связи между блоками, основываясь не только на точке зрения разработчика, но и на представлениях пользователей. Дело в том, что самый распространенный способ поиска, а именно поиск по классификации признаков, работает только в том случае, когда пользователи согласны с принципами этой классификации. Большинство же понятий однозначно классифицированы быть не могут из-за наличия слишком большого количества значимых признаков.

Из этой ситуации есть выход. Существует простой способ классификации – способ карточной сортировки. Все понятия, которые требуется классифицировать, пишутся на карточках. После чего группе пользователей из целевой аудитории предлагается эти карточки рассортировать (при этом каждый субъект получает свой набор карточек).

Получившиеся кучки из карточек нужно разобрать на составляющие и свести результаты от разных субъектов в один способ классификации.

Процессуальная связь. В результате наблюдения за пользователем можно выявить типовые последовательности его действий и жестко реализовать эти последовательности в интерфейсе. Классическим примером жестко заданной процессуальной связи является устройство Мастеров, при котором пользователя заставляют нажимать кнопку Далее.

Проектирование навигационной системы

На основе разработанной ранее структуры экранов на этом этапе выбирается наиболее адекватная навигационная система и разрабатывается её детальный интерфейс.

Навигационная система показывает механизм распределения функций и задач между окнами программы. Навигационная система определяет, каким образом пользователи смогут перемещаться как между различными задачами, так и внутри отдельной задачи. Например, можно ли будет оставить частично завершенную задачу и начать другую.

Проектирование структуры справочной системы

Разрабатывается справочная система (уже известна вся информация, необходимая для выработки ее структуры).

Проектирование экранов

На данном этапе разрабатываются интерфейсы основных, а затем второстепенных экранов системы (к ним относятся диалоговые окна и всевозможные сообщения).

4.4.3. Построение прототипа ПИ

Создание максимально эффективного прототипа интерфейса является чрезвычайно важной задачей. Прототип должен хорошо выглядеть, чтобы понравиться заказчику, он должен быть максимально дешев, максимально полон и, что немаловажно, должен с легкостью обновляться.

Существуют четыре версии прототипа. Особый интерес представляют первые две. Третья и четвертая используются редко, т.к. не сильно отличаются от готовой программы.

- Бумажная
- Презентационная
- Псевдореальная

- Реальная

Первая версия. Бумажная

Необходимо нарисовать на бумаге все экраны и диалоговые окна (или распечатать соответствующие части схемы). Эта распечатка и является первым прототипом. На нём вполне можно тестировать восприятие системы пользователем и её основную логику.

Польза начального прототипирования на бумаге заключается, во-первых, в простоте и скорости рисования, во-вторых, в исключительной простоте модификации по результатам тестирования, а в-третьих, в относительной простоте привлечения представителей целевой аудитории. Значительно легче завлечь субъекта к письменному столу, нежели к компьютеру. Кроме того, бумага помогает не думать о том, как интерфейс выглядит, позволяя сосредоточиться на том, как интерфейс работает.

Вторая версия. Презентационная.

После исчерпания возможностей бумажной версии прототипа стоит создать новую версию. Для этого точно так же рисуется интерфейс, но уже не на бумаге, но в какой-либо презентационной программе. С этой версией прототипа можно тестировать значительно более сложное взаимодействие человека с системой. С другой стороны, исправление найденных ошибок значительно более трудоемко.

Самым распространенным инструментом для создания прототипов второго типа является MS Visio. Некоторую конкуренцию ему могут составить MS PowerPoint и Macromedia FreeHand (и вообще любой иллюстративный пакет, обладающий возможностью работать с несколькими страницами), но возможности PowerPoint для такой работы слишком малы, а возможности FreeHand, напротив, слишком велики.

При работе с Visio можно выбрать одну из двух возможностей: можно либо рисовать все экраны на одном листе, соединяя взаимосвязанные элементы управления и экраны линиями, либо рисовать каждый экран на отдельном листе, связывая экраны ссылками. Первый вариант хорош для вас, поскольку позволяет оценить интерфейс в целом, второй – для заказчика, поскольку его легче понять. Как правило, превратить второй вариант в первый оказывается гораздо легче. Если рисовать в PowerPoint, каждой экран удобно создавать как отдельный слайд, а результат нажатия кнопок имитировать переходами между слайдами.

У многих разработчиков, знакомых с программированием, есть соблазн воспользоваться для создания прототипа так называемыми

«средствами быстрой разработки приложений» (Delphi, Visual Basic и т.д.). Это значительно замедлит процесс создания прототипа, при этом созданный таким образом прототип будет неудобен для быстрой модификации вместе с заказчиком.

Третья версия. Псевдореальная.

В тех случаях, когда в интерфейсе появляются нестандартные элементы или необходимо проверить реальную скорость взаимодействия пользователя с системой, создается еще одна версия прототипа – реально выглядящая, но лишенная каких-либо алгоритмов и, соответственно, не показывающая реальных данных. Делать этот вариант можно как в средах разработки, благо в них есть визуальные инструменты создания интерфейсов, так и в редакторах изображений, что обычно быстрее. Фактически при этом создаются фальшивые снимки экрана, на которых и производят тестирование.

Четвертая версия. Реальная.

Иногда необходимо тестировать взаимодействие пользователя не только с интерфейсом системы, но и с обрабатываемыми системой данными.

Понятно, что создание прототипа в таких условиях не поможет, поскольку прототип вообще не будет отличаться от проектируемой системы. В таких условиях лучше всего написать нужные участки кода до написания всего остального и проводить тестирование уже на реальной системе.

4.4.4. Юзабилити-тестирование

Юзабилити-тестирование представляет собой постановку экспериментов с целью выявления специфичной информации, касающейся дизайна исследуемого продукта. Всё, что для этого нужно, это несколько пользователей средней квалификации, никогда не видевшие тестируемой системы, и ее прототип (на заключительном этапе — реальная версия).

Юзабилити-тестирование производится на протяжении всего цикла разработки продукта. На ранних стадиях разработки тестирование предыдущей версии или конкурирующих продуктов позволяет наметить контрольные точки, которых необходимо достигнуть в процессе разработки. В середине работы над продуктом тестирование предоставляет обратную связь, сообщая места, где дизайн нуждается в улучшении. На заключительных этапах тестирование удостоверяет, что продукт соответствует тем целям, для которых был спроектирован.

Методики тестирования

Рассмотрим некоторые из методик, применяемых на разных этапах.

Метод фокусных групп заключается в опросе специально отобранной группы пользователей. В исследование, которое обычно продолжается около 2 часов, вовлекается от 6 до 9 пользователей. Основное достоинство фокусных групп состоит в том, что они позволяют выявлять спонтанные реакции и идеи и оценивать отношение к этим идеям группы в целом. Результаты работы фокусной группы заносятся в специальный протокол для дальнейшей обработки.

Метод фокусных групп подходит для того, чтобы быстро получить диапазон мнений и оценок пользователей по поводу тех или иных вещей. Обычно этот метод применяют на ранних этапах, еще до начала непосредственной разработки. Конечно, этот метод можно применять и позже, например, для уточнения деталей или при пользовательском тестировании.

Проверка посредством наблюдения за пользователем - один из самых простых и эффективных видов тестирования. Пользователю дается задание, он его выполняет, его действия фиксируются для дальнейшего анализа на камеру, либо какой-нибудь программой записи состояния экрана (например, Lotus ScreenCam).

Метод исключительно полезен для выявления неоднозначности элементов интерфейса. Поскольку каждая неоднозначность приводит к пользовательской ошибке, а каждая такая ошибка фиксируется, обнаружить их при просмотре записанного материала очень легко. Кроме того, если замерять время выполнения задания (секундомером), можно оценить производительность работы пользователей.

Метод используется на всех стадиях разработки.

Мыслим вслух. В течение теста, пока участник выполняет то или иное задание в рамках своего сценария, его просят произносить вслух все мысли, возникающие в процессе взаимодействия с продуктом. Комментарии записываются на диктофон или камеру, а затем анализируются. «Мысли вслух» позволяют понять, как пользователь подходит к интерфейсу и какими соображениями он руководствуется, используя этот интерфейс.

Это недорогой путь получения хорошей качественной ответной реакции на любой стадии разработки.

Методика GOMS

В 1983 году Кард, Моран и Ньювел создали метод оценки скорости работы с системой, названный аббревиатурой GOMS (Goals,

Operators, Methods, and Selection Rules – цели, операторы, методы и правила их выбора).

Идея метода очень проста: все действия пользователя можно разложить на составляющие (например, взять мышь или передвинуть курсор). Ограничив номенклатуру этих составляющих, можно замерить время их выполнения на массе пользователей, после чего получить статистически верные значения длительности этих составляющих.

С помощью тщательных лабораторных исследований был получен набор временных интервалов, требуемых для выполнения различных действий, приведенный в таблице 4.2.

Таблица 4.2

Тип	Действие	Время, с	Комментарии
К	Нажатие на клавишу клавиатуры	0,28	Включая клавиши Alt, Ctrl, Shift
М	Нажатие на кнопку мыши	0,1	
П	Перемещение курсора мыши	1,1	Время, затрачиваемое на перемещение курсора, зависит как от дистанции, так и от размера цели. Тем не менее это достаточно точный компромисс.
В	Перемещение руки с мыши на клавиатуру или наоборот.	0,4	
Д	Ментальная подготовка.	1,2	Время, необходимое пользователю для того, чтобы умственно подготовиться к следующему шагу.
Р	Время реакции системы	От 0,1 до бесконечности	Для базовых операций, таких как работа с меню, это время можно не засчитывать.

На практике указанные значения могут варьироваться в широких пределах. Для опытного пользователя, способного печатать со скоростью 135 слов/мин., значение К может составлять 0,08 с, для обычного пользователя, имеющего скорость 55 слов/мин., - 0,2 с, для сред-

него неопытного пользователя, имеющего скорость 40 слов/мин., - 0,28 с, а для начинающего – 1,2 с.

Скорость набора зависит и от того, что именно набирается. Для того чтобы набрать одну букву из группы случайно взятых букв, большинству людей требуется около 0,5 с. Если же это какой-то запутанный код (например адрес электронной почты), то у большинства людей скорость набора составит около 0,75 символов в секунду.

Более сложным является определение точек, в которых пользователь остановится, чтобы выполнить бессознательную ментальную операцию, - интервалы ментальной подготовки Д. Вот основные правила, позволяющие определить, в какие моменты будут проходить ментальные операции:

- Операторы Д следует устанавливать перед всеми операторами М, предназначенными для выбора команд.
- Если оператор, следующий за оператором Д, является полностью ожидаемым, то этот оператор Д может быть удален. Например, если вы перемещаете мышь с намерением нажать ее кнопку по достижении цели движения, то в этом случае последовательность ВДМ превращается в ВМ.
- Операторы Д следует устанавливать перед всеми операторами К. Но если строка вида ДКДКДК принадлежит когнитивной единице, то следует удалить все операторы К, кроме первого. Когнитивной единицей является непрерывная последовательность вводимых символов, которые образуют одно слово или число.
- Если оператор К является разделителем, стоящим после постоянной строки (например, название команды или любая последовательность символов, которая каждый раз вводится в неизменном виде), то следует удалить оператор Д, стоящий перед ним. Но если оператор К является разделителем для строки аргументов или любой другой изменяемой строки, то оператор Д следует сохранить перед ним.
- Любую часть оператора Д, которая перекрывает оператор Р, учитывать не следует.

Широкая изменяемость каждой из представленных метрик объясняет, почему эта упрощенная модель не может использоваться для получения абсолютных временных значений с какой-либо степенью точности. Тем не менее, с помощью типичных значений мы можем сделать правильную сравнительную оценку между какими-то двумя интерфейсами.

К сожалению, этот метод имеет недостатки:

- он применим в основном для предсказания действий опытных пользователей;
- он никак не учитывает ни прогресса в обучении, ни возможных ошибок, ни степени удовлетворения пользователей;
- он плохо применим при проектировании сайтов из-за непредсказуемого времени реакции системы.

Тем не менее, это самый дешевый метод, который вообще не требует участия испытуемых, поэтому можно рекомендовать его на стадиях проектирования и реализации интерфейса.

ЗАКЛЮЧЕНИЕ

Рассмотренные в данных методических указаниях WIMP-интерфейсы в настоящее время являются стандартом де-факто при проектировании пользовательских интерфейсов разнообразных программных продуктов. Однако эта область развивается столь стремительно и динамично, что трудно строить прогнозы относительно интерфейсов нового поколения, которые, безусловно, придут им на смену. Тем не менее, те базовые принципы человеко-машинного взаимодействия и общие рекомендации по созданию эргономичных ПИ, которые здесь изложены, справедливы вне зависимости от конкретной реализации интерфейса.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Боресков, А. Компьютерная графика/А.В.Боресков, Е.В. Шикин, Г.Е.Шикина – М.:Финансы и статистика,1996. – 176 с.
2. Титтел, Э. Создание VRML миров / Э. Титтел, К. Сандерс, Ч. Скот, П. Вольф: Пер. с англ. – Киев: Издательская группа BHV, 1997. – 320 с.
3. Приписнов, Д. Моделирование в 3D Studio MAX 3.0 / Д.Ю.Приписнов – СПб.: БХВ – Санкт-Петербург, 2000. – 352 с.
4. Картузов, А. Человеко-машинное взаимодействие (интерактивные графические системы) [Электронный ресурс] / А.В.Картузов – Режим доступа: htpp://coop.chuvashia.ru/SanyaSoft/HCI/banner.htm
5. Кирсанов, Д. Веб-дизайн / Д.Кирсанов – СПб.: Издательство «Символ», 2002. – 376 с.
6. Головач, В. Дизайн пользовательского интерфейса [Электронный ресурс] / В.Головач – Режим доступа: htpp://www.uibook.ru
7. Международный стандарт ISO DIS 9241-11. Эргономические требования к офисной работе с визуальными терминалами (VDTs). Дата введения – сентябрь 1994г. [Электронный ресурс] – Режим доступа: http://www.hci.ru/standards.htm

Подписано в печать 31.08.2011.	Усл. печ. л. 1,7	Тираж экз
Печать офсетная.	Бумага писчая.	Заказ №____.

Отпечатано: РИО ВоГТУ, г. Вологда, ул. Ленина, 15