Тема 1

Введение в проектирование пользовательских интерфейсов

1. Характеристика пользовательского интерфейса
   1.1. Интерфейс, его виды.
   1.2. Основные функции пользовательского интерфейса и требования к нему.
   1.3. Стили пользовательского интерфейса.
2. Проектирование пользовательского интерфейса.
   2.1. Значение проектирования пользовательского интерфейсапри создании социотехнических систем.
   2.2. Принципы создания интерфейса.
   2.3. Количественный анализ интерфейса.

1. Характеристика пользовательского интерфейса

1.1. Интерфейс, его виды.

Интерфейс – это совокупность средств и методов обеспечения взаимодействия между элементами системы. В зависимости от выполняемых функций выделяют следующие основные виды интерфейса: *интерфейс передачи данных, программный интерфейс, аппаратный интерфейс и интерфейс пользователя или пользовательский интерфейс (ПИ)*.

Интерфейс передачи данных –средства и методы передачи данных между устройствами. В зависимости от способа передачи данных различают последовательный и параллельный интерфейсы.

Программный интерфейс — система унифицированных связей, предназначенных для обмена информацией между компонентами вычислительной системы. Программный интерфейс задает набор необходимых процедур, их параметров и способов обращения.

Аппаратный интерфейс – комплекс технических средств, преобразующих сигналы и передающих их от одного технического компонента системы к другому. Физически аппаратный интерфейс определяется набором электрических связей и характеристиками сигналов.

Интерфейс пользователя или пользовательский интерфейс (ПИ) –это комплекс технических и информационно-программных средств, посредством которых осуществляется взаимодействие человека и технического звена эргатической системы; совокупность средств и

методов, позволяющих пользователю управлять работой технического звена социотехнической системы (программой, устройствами, аппаратурой) и получать требуемые результаты.

К техническим средствам ПИ относятся средства отображения информации (СОИ) и органы управления (ОУ), используемые человеком при осуществлении диалога с технической системой: клавиши, переключатели, тумблер, мышь, джойстик, сенсорный экран, дисплей, индикатор и т.п. Программные средства интерфейса — совокупность программных средств, обеспечивающих диалог человека с вычислительными средствами и визуализацию виртуальных объектов на экране.Результатом отображения информации является *информационная модель* — условное отображение, информация о состоянии объекта воздействия, системы «человек – машина» и способов управления ими.

ПИ включает

1. визуальное оформление, отвечающее за представление информации оператору;
2. функциональные возможности системы;
3. техники взаимодействия (диалога) оператора с системой.

Выделяют аппаратные и виртуальные средства диалога ПИ. Аппаратные средства диалога технически поддерживают человеко-машинное взаимодействие в системе «человек – компьютер». Примером аппаратных средств являются

- дисплей – используется для визуального отображения отклика компьютерной системы на действия пользователя;
- клавиатура, с ее помощью пользователь может передавать компьютеру данные или команды;
- мышь – дает возможность быстро выбирать решения из вариантов, манипулируя свойствами объектов, отображаемых на дисплее.
- джойстик;
- мультимедиа-устройства – средства голосового взаимодействия, сенсорного ввода и т.п.

К виртуальным устройствам диалога относятся реально несуществующие устройства, поддерживаемые специальными программными средствами, например, тренажеры.

Выделяют несколько видов структуры диалога ПИ.

1. Диалог типа "вопрос-ответ". Структура такого диалога аналогична интервью. Система задает пользователю вопросы и получает от него ответ (информацию). Форма задания вопроса и ввода ответа может быть разной (списки, поля ввода значений и др.).

2. Диалог на основе меню. Система предъявляет пользователю перечень возможных решений в виде иерархически организованных списков, в которых пользователь выбирает

нужное.

3. Диалог на основе экранных форм. Такая структура позволяет на одном шаге получить от пользователя много данных, указываемых в форме. В таких формах могут присутствовать списки выбора, поля ввода и др.

4. Диалог на основе командного языка. Часто используется в операционных системах для прямого ввода команды в командной строке.

1.2. Основные функции пользовательского интерфейса и требования к нему[1]

«Любой пользовательский интерфейс (ПИ) должен обеспечивать выполнение следующих четырёх основных функций:

2. Управления компьютером путём действий оператора (пользователя): инициация, прерывание, отмена компьютерных процессов и т. п.

3. Ввод данных, осуществляемый оператором, и отклик системы.

4. Отображение данных, включающее отображение данных, вводимых оператором, который может управлять процессом отображения данных.

5. Поддержка оператора в процессе деятельности, осуществляемая по каналам обратной связи, в которых циркулирует информация об ошибочных или случайных (не по алгоритму) действиях оператора.

Эффективный ПИ должен обеспечивать всестороннее использование потенциальных возможностей оператора, технических и информационно-программных средств АРМ, высокие безошибочность и быстродействие деятельности оператора при применении ПИ по назначению. Хорошо спроектированный ПИ обеспечивает максимальный комфорт деятельности оператора и должен:

- способствовать быстрому освоению вычислительной техники оператором, формированию у него устойчивых навыков;
- быть спроектирован таким образом, чтобы оператор вводил информацию естественным образом, не заботясь о ходе вычислительного процесса;
- удовлетворять рабочие потребности человека-оператора, а не обслуживать процесс обработки данных.

---

1. Сергеев, С. Ф., Введение в проектирование интеллектуальных интерфейсов: Учебное пособие. / С.Ф. Сергеев, П.И. Падерно, Н.А. Назаренко – СПб: СПбГУ ИТМО, 2011. – С. 25-26.

Синтаксическая структура, реализованная в интерфейсе должна:

- соответствовать ожиданиям оператора решающего профессиональную задачу;
- содержать систему правил работы оператора, обеспечивающую лёгкое управление системой;
- постоянно находиться под контролем оператора, никакие действия последнего не должны приводить к тупиковой ситуации или зависанию программы.

Интерфейс и справочный механизм информационной системы должны:

- обеспечивать возможность лёгкого исправления ошибок ввода, не должен требовать повторного ввода данных;
- обеспечивать оператора информацией, позволяющей ему управлять диалогом, распознавать и исправлять ошибки, определять последующие действия, входящие в алгоритм;
- обеспечить конкретность и понятность оператору выдаваемой компьютером информации;
- согласование объёма представляемой оператору информации с возможностями его кратковременной памяти;
- в информации об ошибках оператора делать акцент не на неправильные действия последнего, а на то, чем и как можно исправить ошибки;
- предусматривать использование четырех видов диалога: меню, команды, манипуляции и диалог посредством заполнения форм.
- при решении каждой задачи предоставить оператору возможность использования не менее двух видов диалога. Критерием выбора вида диалога в ходе решения конкретной задачи является обеспечение заданных показателей безошибочности и быстродействия.»

1.3. Стили пользовательского интерфейса[2]

«Существует ряд стилей пользовательского интерфейса, которые завоевали популярность в индустрии программных средств. Основные виды стилей ПИ представлены на рисунке 1.

Самыми популярными являются *GUI*-интерфейсы (*GUI – Graphical User Interface*) и

---

2. Сергеев, С.Ф., Введение в проектирование интеллектуальных интерфейсов: Учебное пособие. / С.Ф. Сергеев, П.И. Падерно, Н.А. Назаренко – СПб: СПбГУ ИТМО, 2011. – С. 27-33.

построенные на их основе *WUI*-интерфейсы (*WUI – Web User Interface*). Стилевые детали *WUI*-интерфейсов незначительно отличаются от *GUI*-интерфейсов, подтверждением чему служат диалоговые окна *Web*-браузеров.

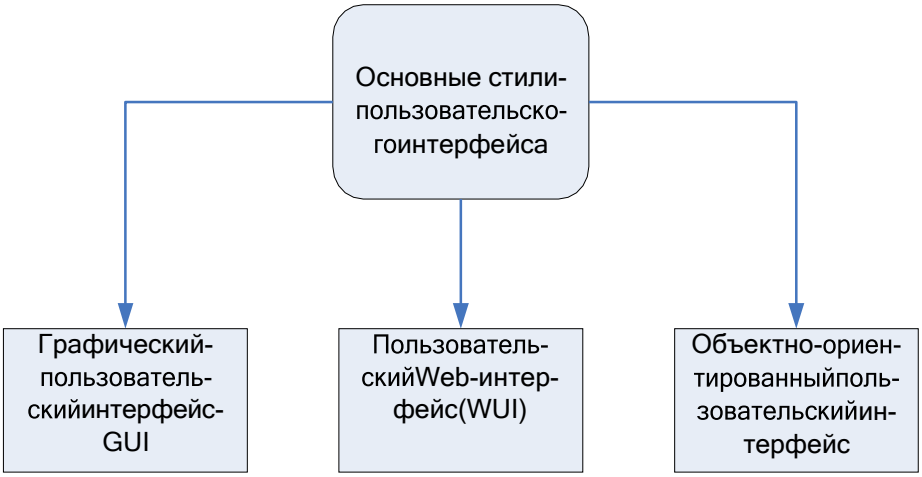


Рис.1. Основные стили пользовательского интерфейса

Графический ПИ (GUI-интерфейс)

Графический пользовательский интерфейс (*Graphical User Interface – GUI*) определяется как стиль взаимодействия «пользователь – компьютер», в котором применяются четыре базовых элемента: окна, пиктограммы, меню и указатели (рис. 2). Иногда *GUI*-интерфейс называют *WIMP*-интерфейсом (*Windows* – окна, *Icons* – пиктограммы, *Menus* – меню и *Pointers* – указатели).

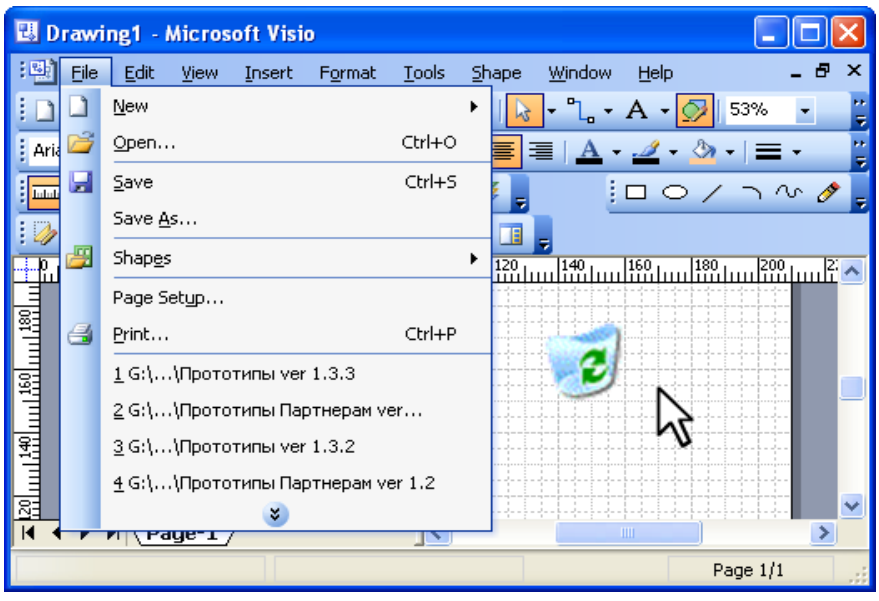


Рис. 2. Пример *GUI*-интерфейса

Важнейшие свойства *GUI*-интерфейса — это возможность непосредственного манипулирования, поддержка мыши или указателя, использование графики и наличие области для функций и данных приложения. Начальный анализ основ *GUI*-стиля ведётся отдельно от прикладного уровня *GUI*-ориентированных приложений.

*Непосредственное манипулирование***.** Наиболее значительное свойство *GUI*-интерфейса заключается в непосредственном манипулировании, которое позволяет пользователю взаимодействовать с объектами с помощью указателя. Например, окно можно переместить по экрану с помощью мыши, установив указатель на строку заголовка окна, нажав и удерживая кнопку мыши и перемещая мышь (иногда эту операцию называют «захватить и перетащить» — *«grab and drag»*). Другой пример непосредственного манипулирования с помощью указателя — это выделение текста («занять [место] и ввести» — *«swipe and type»*) или рисование непосредственно в графической области с использованием указателя и графических инструментов наподобие кисти (*paint brush*).

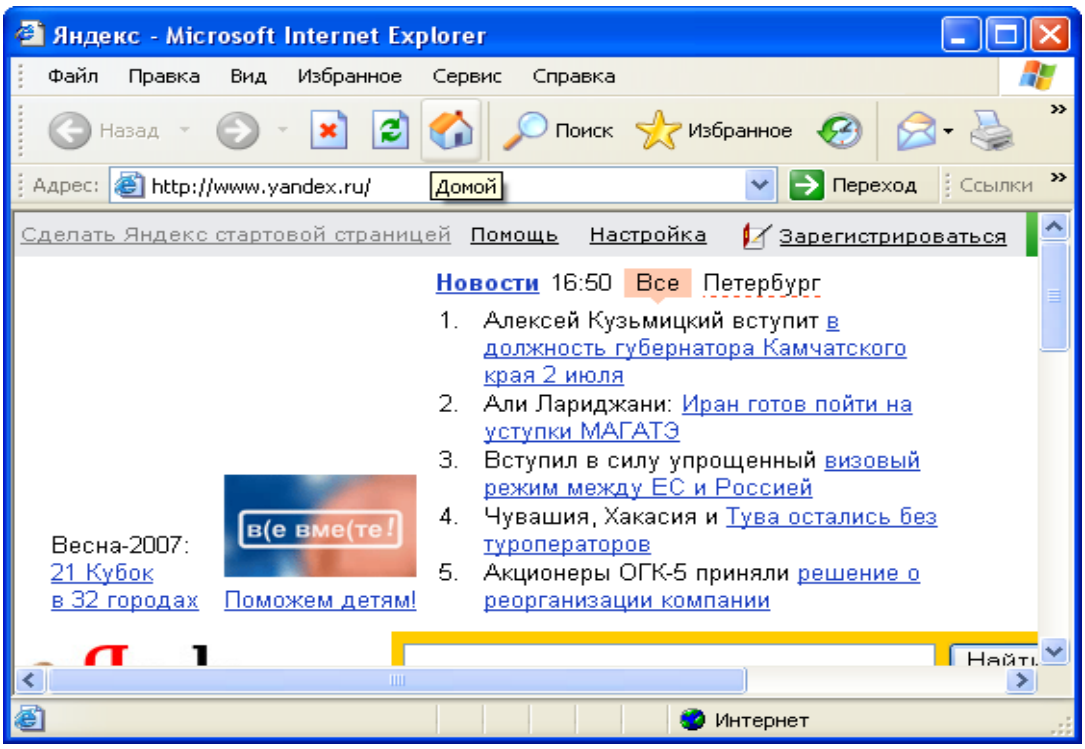
Многие действия, выполняемые с помощью выбора альтернатив или меню, можно произвести, воспользовавшись непосредственным манипулированием. Например, во многих системах результатом перетаскивания пиктограммы документа на пиктограмму принтера на рабочем столе является печать документа. К другим действиям, которые выполняются с помощью непосредственной манипуляции, относятся такие операции, как *Move* (Переместить), *Copy* (Копировать), *Delete* (Удалить) и *Link* (Связать).

*Другие свойства.*К некоторым другим методам работы ПИ, присущим *GUI*-интерфейсу, относятся буфер обмена, комбинации клавиш, ускоряющие клавиши в меню и диалогах, а также дополнительные возможности взаимодействия мышь-клавиатура. Несмотря на свою полезность, эти механизмы не рассматриваются как существенные свойства *GUI*-интерфейса.

Пользовательский Web-интерфейс (WUI-интерфейс)

Базовый *WUI*-стиль (*Web User Interface*) весьма схож с меню иерархической структуры, которое пользователи знают по опыту работы в средах с графическим интерфейсом за исключением более наглядного представления и использования гиперссылок. Необходимая навигация выполняется в рамках одного или нескольких приложений с использованием текстовых или визуальных гиперссылок. В зависимости от структуры гиперссылок приложения навигация в пределах *WUI*-интерфейса приводит к отображению *Web*-страниц в иерархии приложенияпо одной за раз — в линейном или нелинейном стиле внутри одного*GUI*-окна. Во многих отношениях *WUI*-ориентированные приложенияэто «шаг назад в будущее»

— или, может быть, нечто худшее, учитывая объёмы электронных документов и других



материалов в формате *Web*.

Рис. 3. Пример *WUI*-интерфейса — *Web*-браузер *Internet Explorer*

Основные особенности приложения, использующего *WUI*-стиль:

- Информация обычно отображается в единственном *GUI*-окне, называемом *Web*-браузером, хотя для представления данных в приложении могут использоваться несколько окон.
- *Web*-браузер обеспечивает меню для *Web*-приложения.
- Выбор действий ограничен, так как меню, обеспечивающее обращение к функциям, не является легкодоступным для приложения.
- *Web*-страница обладает небольшой степенью внутреннего контроля над клиентской областью для открытия специализированных всплывающих меню.
- Создание специализированных меню требует дополнительной работы по программированию.
- Функциональные возможности приложения должны отображаться в методы для вызова команд.
- Клиентская область не содержит традиционные пиктограммы.
- Многие приложения используют графику и анимацию в эстетических или навигационных целях. Это таит в себе потенциальную угрозу возникновения внешнего визуального

шума и увеличения времён отклика при загрузке и раскрытии графических файлов.

• *Web*-браузер и приложения обеспечивают возможности отключения графики, содержащейся в *Web*-страницах, так что на экране отображается только их текстовая версия.

• Поддержка указателя осуществляется в основном для выборас помощью одного щелчка мышью или выбора по навигационным ссылкам. Технология *«drag and drop»* («перетащить и поместить») не поддерживается за исключением случаев специального программирования в определенных средах.

Web-ориентированное ПО становится все более похожим на *GUI*- ориентированное ПО (возможно потому, что пользователи неизменно требуют наличия популярных и полезных свойств *GUI*-интерфейса наподобие функции *«drag and drop»* или всплывающих меню).

Навигация. Переход от одной страницы к другой с использованием гиперссылок или поискового механизма — наиболее часто выполняемая функция *WUI*-интерфейса. Страницы, с которыми встречается пользователь, существуют в пределах того же самого или другого Web- узла.

Web-браузер обеспечивает базовые возможности навигации для перемещения по *Web*-узлам и в пределах *Web*-узлов линейным способом с помощью кнопок панели инструментов *Back* (Назад) и *Forward* (Вперед). Навигация от одной страницы приложения к другой в пределах одного и того же *Web*-узла приложения выполняется с использованием гиперссылок, схемы *Web*-узла, кнопок и навигационной панели.

Представление и поведение. Основное назначение *Web*-страницы заключается в обеспечении полезной информацией, включая навигационную структуру и организацию *Web*-узла. *Web*-страницы составлены из одной или нескольких конструкций, представляющих собой сочетание мозаик цветных графических элементов. По сравнению с *GUI*-ориентированными приложениями *WUI*-ориентированные приложения включают большое количество элементов поведения, которые не вызываются пользователем, например, анимационных. С точки зрения ПИ в *Internet* царит полная анархия.

Компоненты WUI-интерфейса. К наиболее распространенным компонентам *WUI*-интерфейса относятся баннеры (заголовки), навигационные панели и визуальные или текстовые гиперссылки, упорядоченные различными способами. Также применяются разнообразные подходы к использованию графики, анимации и цвета.

• *Баннер* представляет собой визуальный заголовок, отображаемый вверху *Web*-страницы.

• *Навигационная панель* — это список вариантов выбора гиперссылок, обеспечивающих доступ к информации сайта.

• *Гиперссылка.* Представляет собой вариант выбора, который отображает следующую

страницу информации или перемещает фокус отображения на другую область той же страницы.

Макеты Web-страниц. Информация представляется на *Web*-страницах с использованием одного или нескольких макетов и навигационных стилей.

- *Браузер*. Типичный браузер обладает заголовком, навигационной панелью и областью, отображаемой в пределах экрана.
- *Каталог*. Каталог представляет собой визуальный поисковый механизм, в котором перечислены варианты выбора гиперссылок, используемых для навигации по дополнительным вариантам выбора до тех пор, пока не будет найден искомый результат. Допускаются навигационные панели в виде заголовков и другие типы навигации по вариантам выбора гиперссылок.
- *Поиск и результаты поиска.* Один или несколько элементов управления, с помощью которых пользователь осуществляет ввод или выбор критерия поиска информации. Результаты поиска отображаются в том же или другом окне *Web*-браузера.
- *Документ.* Во многом похожий на свой бумажный двойник*Web*-документ отображает текстовую информацию вместе со ссылкамина дополнительные источники или развернутое представление информации.
- *Записная книжка*. Некоторые *Web*-узлы представляют визуальную записную книжку в качестве метафоры для организации данных. Она почти не отличается от навигационной панели, с той лишь разницей, что содержит меньшее количество вариантов выбора.

Сбор данных. Дополнительной целью некоторых *Web*-страниц является сбор имен и адресов, а также другой информации о пользователях. Для этого используются обычные элементы управления *GUI*-интерфейса, иногда с некоторыми ограничениями.

Проблемы проектирования. Факторы успешного проектирования, которые влияют на качество приложений, использующих *Web*-стиль ПИ: простота навигации по иерархическим информационным структурам, лёгкость и быстрота поиска, а также быстрая реакция. К другим важным факторам относятся эстетические характеристики и ценность текущего содержания информации.

Объектно-ориентированный пользовательский интерфейс

Проектирование программных объектов даёт возможность предоставить в распоряжение пользователя приложение, обладающее объектно-ориентированным стилем ПИ и/или объектно-ориентированной внутренней структурой (реализацией). Многие объектно-ориентированные свойства реального мира находят отражение во внешнем виде, поведении,

требованиях к взаимодействию и функциональных возможностях. Компьютеризованное усовершенствование или дополнение объектов реального мира, если только оно плохо спроектировано или реализовано, не очевидно для конечного пользователя и не в состоянии преодолеть его устоявшиеся знания и восприятие. Представленные в явном виде при проектировании, обозначения классов объектов, иерархии классов и наследование посредством иерархии классов остаются прозрачными для пользователя.

Объектно-ориентированный прикладной ПИ должен обладать следующими свойствами:

- обеспечивать непосредственное манипулирование (перетаскивать любые объекты куда угодно);
- обеспечивать непосредственный ввод данных (записывать любую информацию);
- обеспечивать контекстную зависимость от объектов (всплывающие (контекстные) меню, справки, согласованность и т. д.).

Хороший прикладной объектно-ориентированный ПИ прост в использовании; это значит, что его механизмы пользовательского интерфейса прозрачны.»

2. Проектирование пользовательского интерфейса

2.1. Значение проектирования пользовательского интерфейса при создании социотехнических систем

Проектирование ПИ – один из самыхважных этапов создания информационной системы. Пользовательский интерфейс является каналом, посредством которого осуществляется взаимодействие пользователя и компьютерной системы. Лучший ПИ – это такой интерфейс, которому пользователь не уделяет много внимания, который в процессе работы почти не замечается пользователем. Разработчики часто рассматривают ПИ отдельно от функциональности системы, а пользователи, как правило, не разделяют функциональность и ПИ: для них ПИ и является программой (сайтом). С точки зрения пользователя, если ПИ хороший, то и сама система приятна в применении, удобна и хороша.

Основные преимущества хорошего ПИ:

1. доступность функциональности системы для максимального количества пользователей и, следовательно, увеличение аудитории продукта;
2. уменьшение потерь продуктивности работников при внедрении системы на предприятии и более быстрое восстановление утраченной продуктивности;
3. снижение затрат на обучение и поддержку пользователей;

4. снижение риска ошибок, совершаемых пользователями;
5. повышение конкурентоспособности продукта.

ГОСТ Р ИСО/МЭК 9126-93 определяет качество любого продукта определяется как «объем признаков и характеристик продукции или услуги, который относится к их способности удовлетворять установленным или предполагаемым потребностям». При комплексной оценке показателей качества программного продукта качество пользовательского интерфейса вносит определяющий вклад в такую субхарактеристику качества, как практичность (usability).

Юзабилити (англ. usability — «возможность использования», «способность быть использованным», «полезность») — степень эффективности, продуктивности и удовлетворенности, с которыми продукт может быть использован определенными пользователями в определенном контексте использования для достижения определенных целей (пункт 3.1 стандарта ISO 9241-11).

«Основная цель юзабилити — проконтролировать, спрогнозировать и воздействовать на процессы создания системы с целью повысить конечную эргономичность продукта.

Уже в самом определении юзабилити присутствуют три важных аспекта:

- определенные пользователи: интерфейс всегда создается для конкретной группы людей (или нескольких групп), характеризующейся своими особенностями, знаниями, навыками, ожиданиями, сильными и слабыми сторонами. Системы никогда не создаются для «любого» или «среднего» пользователя, даже если позиционируются как «подходящие всем»;
- определённый контекст использования: решения, которые подходят для пользователя идеально в одном контексте, могут быть совершенно неприемлемы в другом. К примеру, управление посредством голосового интерфейса может оказаться удачным решением в условиях работы оператора в изолированном помещении, однако для совместной работы в одной комнате использование такой системы несколькими операторами одновременно будет затруднительным;
- определённые цели: ни одно решение в принципе не может быть удачным, если оно не помогает достичь пользователю его целей. А для того, чтобы оно помогало их достичь, нужно узнать, в чем эти цели состоят.

Интерфейс должен быть юзабильным. Но не бывает абсолютной вещи, удобной для всех. Она удобна сейчас для решения конкретной задачи конкретным человеком, но все изменяется, как только здесь изменится какая-то переменная, т. е. другой человек, не тот, на которого рассчитывали (например, рассчитывали на мужчин, а пришла девушка или наоборот, сделали для девушки, а пришел инженер-мужчина — и не соответствует продукт

его ментальным моделям). Он пытается не по назначению использовать продукт. Например, сделали веб-сайт, а он его пытается с КПК или мобильного телефона смотреть. Но сайт не рассчитан на это.» [3]

Выделяют следующие основные принципы создания интерфейса

1. Естественность (интуитивность). Работа с системой не должна вызывать у пользователя сложностей в поиске необходимых элементов интерфейса для решения поставленной задачи.

2. Непротиворечивость. Если в процессе взаимодействия с одной частью системы пользователем были использованы определенные приемы работы, то при работе с другой частью системы приемы работы должны быть идентичны. Кроме того, работа ПИ системы должна соответствовать установленным привычным нормам (например, использование клавиши «Escape»).

3. Неизбыточность, т.е. пользователь должен вводить минимальную информацию для работы или управления системой. Например, пользователь не должен вводить незначимые цифры (00010 вместо 10) или повторно вводить информацию, которая уже была ранее введена или которая может быть автоматически получена из системы.

4. Непосредственный доступ к системе помощи – подсказкам и инструкциям. Сообщения об ошибках должны быть полезны и понятны пользователю

5. Гибкость – возможность использования системы пользователями с различными уровнями подготовки. Например, для начинающих пользователей интерфейс может быть организован как иерархическая структура меню, а для опытных – как команды, комбинации нажатий клавиш и т.п.

2.2. Критерии качества пользовательского интерфейса

Выделяют следующие основных критерии качества ПИ:

1. скорость работы пользователей;
2. количество совершаемых человеком ошибок;
3. скорость обучения работе с системой;
4. субъективное удовлетворение пользователей.

---

3. Брусенцова, Т. П. Проектирование интерфейсов пользователя: пособие для студентов специальности 1-47 01 02 «Дизайн электронных и веб-изданий» / Т. П. Брусенцова, Т. В. Кишкурно. — Минск БГТУ, 2019. — С. 12-13.

При этом соответствие интерфейса задачам пользователя рассматривается как неотъемлемая характеристика ПИ.

Рассмотрим эти критерии более подробно.

Скорость работы пользователей

Скорость выполнения работы является важным критерием эффективности интерфейса, т.к. влияет на производительность труда. Время, необходимое пользователю на работу с системой, включает следующие компоненты:

- прием информации (например, считывание показаний);
- оценка и переработка информации;
- принятие решения;
- реализация решения;
- реакция системы.

К сожалению, существенно повысить скорость протекания когнитивных процессов через изменение интерфейса практически невозможно. Но при проектировании ПИ можно уменьшить влияние факторов, усложняющих и, следовательно, замедляющих процесс мышления. Рассмотрим основные факторы, позволяющие ускорить процесс мышления.

1. Непосредственное манипулирование. Вместо того, чтобы отдавать команды системе, пользователь манипулирует объектами. Например, пользователь ОС Windows перетаскивает пиктограмму файла в корзину на рабочем столе, вместо ввода команды «удалить».

2. Применение в ПИ эффективных методов при потере фокуса внимания. В процессе работы с системой пользователи отвлекаются. Следовательно, необходимо облегчать их возвращение к работе. Для этого пользователь в любой момент должен знать:

- на каком шаге он остановился;
- какие команды и параметры он уже дал системе;
- что именно он должен сделать на текущем шаге;
- куда было обращено его внимание на момент отвлечения.

Предоставлять пользователю всю эту информацию лучше визуально.

Длительность физических действий пользователя зависит от степени автоматизации работы и требований к точности работы. Степень автоматизации зависит от того, как распределены функции между человеком и компьютером. Чем выше требования к точности, тем меньше скорость работы. Так, клавиатура, в отличие от манипулятора «мышь», не требует большой точности движений. Мышь не предназначена для очень точных, в 1 или 2 пикселя

движений. Следовательно, наличие в ПИ элементов малого размера будет снижать скорость работы пользователя, если он применяет мышь. Повысить скорость работы можно путем оптимизации использования мыши. Чтобы ускорить клик мышью кнопки в ПИ, ее можно сделать бесконечного размера и (или)сделать нулевой дистанцию до нее от курсора (например, как контекстное меню, вызываемое правой кнопкой мыши).

Количество совершаемых человеком ошибок

Выделяют следующие типы ошибок.

1) *Ошибки, вызванные недостаточным знанием предметной области.*Для уменьшения количества таких ошибок необходимо обучать пользователей.

2) *Опечатки.* Онипоявляются в двух случаях.

- Когда выполнению текущего действия уделяется недостаточно внимания. Это типично, прежде всего, для опытных пользователей, не проверяющих каждое свое действие.
- Когда в мысленный план выполняемого действия вклинивается фрагмент плана другого действия. Как правило, это происходит в тех случаях, когда пользователь, выполняя текущее действие, одновременно продумывает следующее действие.

3) *Ошибки, вызванные несчитыванием показаний системы*. Опытные пользователи не всегда считывают показания системы, потому что у них уже сложилось мнение о текущем состоянии, и они не считают нужным его проверять. Неопытные пользователи порой забывают считывать показания или не знают, что это нужно делать и как. Поэтому важные показатели нужно делать более заметными, чтобы они привлекали внимание пользователя.

4) *Моторные ошибки,* когда пользователь знает, что и как он должен сделать, но необходимые физические действия трудно выполнить. Например, моторные ошибки с большой вероятностью возникнут, если необходимо нажать на кнопку очень маленького размера. Основнойспособ как избежать этих ошибок – снижение требований к точности движений пользователя.

Итак, для минимизации количества ошибок нужнообучать пользователей в процессе работы, повышать разборчивость и заметность индикаторов состояния и снижать чувствительность системы к ошибкам. Чтобы снизить чувствительность системы к ошибкам целесообразно при проектировании предусмотреть

- блокировку потенциально опасных действий пользователя до получения подтверждения правильности действия;
- проверку системой всех действий пользователя перед их принятием;
- самостоятельный выбор системой необходимых команд или параметров, когда от пользователя требуется только проверка.

Скорость обучения

Пользователь будет учиться как работать с системой, как выполнить какую-либо функцию, только если он знает о ее существовании, и считает, что он сможет получить от этого выгоду (пользу).Выделяют следующие основные способы повышения эффективности обучения работе с системой: бумажная документация, обучающие материалы и общая «понятность» системы. Понятность системы включает четыре составляющие: ментальную модель, метафору, аффорданс и стандарт.

Обучающие материалы

К обучающим материалам прежде всего относятся различные подсистемы справки: базовая справка, обзорная справка, справка предметной области, процедурная справка, контекстная справка и справка состояния.

Базовая справка объясняет пользователю сущность и назначение системы. Обычно предъявляется только один раз. Обзорная справка описывает функции системы, она также очно предъявляется только один раз. Если функциональность системы велика, невозможно добиться того, чтобы пользователи запоминали обзорную справку за один раз. В этом случае оптимальным вариантом является отслеживание действий пользователя и в случае заранее определенных действий пользователей предъявление коротких фраз типа «А Вы знаете, что...». Справка предметной области подсказывает ответ на вопрос «Как сделать хорошо?», а процедурная – на вопрос «Как это сделать?». Контекстная справка отвечает на вопросы «Что это делает?» и «Зачем это нужно?». Как правило, пользователи обращаются к контекстной справке во время выполнения какого-либо действия, поэтому она не должна прерывать это действие и содержать минимальный объем информации. Справка состояния отвечает на вопрос «Что происходит в настоящий момент?».

Ментальная модель – когда пользовательоднозначно понимает, для чего и как можно использовать определенный предмет, устройство (систему), как оно работает, при этом он может не знать сущности происходящих в устройстве (системе) процессов.

Метафора. Метафора используется в графических интерфейсах, чтобы подсказать пользователю, как функционирует компьютерная система и для чего она предназначена.

Метафора позволяет пользователю не создавать новую ментальную модель, а воспользоваться готовой, которую он ранее построил по другому поводу. Например, метафора рабочего стола (desktop), применяемая в Windows и Macintosh, где изображения папок на экране компьютера напоминают реальные папки, которые используются для хранения и сортировки документов в офисе.

Аффорданс – ситуация, когда объект показывает человеку способ своего использования своими неотъемлемыми свойствами. Глядя на аффорданс, пользователь понимает, какое

действие требуется совершить, и что после этого произойдет. Аффорданс позволяет пользователям работать с определённым элементом системы без предварительного обучения. Применительно к разработке ПИ впервые понятие аффорданса – подсказки или указателя на функциональность того или иного элемента – начал использовать Дональд Норман**.** Примером аффорданса являются стрелки выпадающего меню или выпуклая кнопка.

Выделяют следующие способы передачи аффорданса:

- повторение конфигурации объектов конфигурацией элементов управления;
- видимая принадлежность управляющих элементов объекту;
- визуальное совпадение аффордансов экранных объектов с такими же аффордансами объектов реального мира (кнопка в реальном мире предлагает пользователю нажать на нее, псевдотрехмерная кнопка – тоже, по аналогии);
- изменение свойств объекта при подведении к нему курсора (аналог тактильного исследования).

Стандарт как средство повышения скорости обучения предполагает унификацию, единообразие подходов, приемов и т.п. При наличии стандартов пользователям необходимо учиться только один раз. Примеромприменения стандарта является дизайн программы FrontPage (которую разрабатывали в Vermeer), похожий на дизайн программ пакета Microsoft Office — Word и Excel. Дизайн программы FrontPage создавался с самого начала так, чтобы она выглядела и работала как приложения Microsoft Office. Дизайн элементов управления, выдержанный в едином стиле для различных программ, помогает пользователю научиться работать с новой программой.

Субъективное удовлетворение пользователей

Основными факторами, которые влияют на субъективное удовлетворение пользователей, являютсясубъективное ощущение эстетики, времени работы, психологического напряжения, собственной глупости, самовыражения.

Субъективное ощущение эстетики(привлекательности) объектов.

Эстетическая привлекательность отражает не красоту или «стиль» приложения, информационной систем, а то, насколько хорошо внешний вид и поведение приложения характеризуют его функции. Так, в приложениях, которые помогают пользователям выполнить серьезную задачу, можно поставить акцент на задаче, сохраняя декоративные элементы тонкими и ненавязчивыми. В то же время в таких приложениях как игра, пользователи ожидают увидеть красочный внешний вид, который обещает получение удовольствия и радости, а не сдержанность и лаконичность.

Субъективное восприятие времени. Любой пользователь хочет работать быстро. Повышение объективной скорости работы зачастую способно повысить и субъективную скорость. Но следует учитывать, что субъективное восприятие времени может отличаться от объективной продолжительности протекания процесса, поэтому методы повышения реальной скорости работы не всегда помогают повысить субъективное удовлетворение пользователей.

Воспринимаемая продолжительность действий зависит от уровня активности пользователя. Известно, что при бездействии время течет очень медленно. Следовательно, субъективную скорость работы можно повысить двумя способами:

- заполнение пауз между событиями (когда пользователь ожидает реакции системы), например, демонстрацией индикатора степени выполнения;
- разделение крупных действий пользователей на более мелкие. При этом количество работы увеличивается, но субъективная длительность уменьшается. Недостатком этого метода является то, что он увеличивает усталость.

Субъективное ощущение психологического напряжения возникает из-за страха совершить ошибку или из-за отсутствия чувства контроля над системой, происходящими процессами. Чтоб уменьшить страх совершить непоправимую ошибку целесообразно предоставить пользователю возможность

- отменять свои предыдущие действия, без ограничения количества уровней отмены и типа отменяемых действий;
- скрывать опасные для пользователя места интерфейса.

Субъективное ощущение собственной глупости как правило, возникает из-за частых и неудачно сформулированных сообщений об ошибках.Идеальное сообщение об ошибке должно отвечать на три вопроса:

1. В чем заключается проблема?
2. Как исправить эту проблему сейчас?
3. Как сделать так, чтобы проблема не повторилась?

Самовыражение.

Пользователи хотят выразить себя, в том числе и через программы, которыми они пользуются. Следовательно, возможность настроить систему под свои нужды является значимым источником субъективного удовлетворения пользователя.

2.3. Количественный анализ интерфейса

Существует несколько методов количественного анализа ПИ. Наиболее известные — это закон Фиттса, закон Хика и модель GOMS, разработанная Кардом, Мораном и Ньюэллом.

В 1954 г. Пол Фиттс (Paul Fitts) сформулировал правило, ставшее известным как Закон Фиттса: «Время достижения цели прямо пропорционально дистанции до цели и обратно пропорционально размеру цели».

$$\text{T дост цели} = a + b \log_2 \left(\frac{D}{S} + 1\right)$$

где a и b устанавливаются опытным путем по параметрам производительности человека. Для практического применения можно принять a=50 и b=150.

D – дистанция от курсора до цели;

S – размер цели по направлению движения курсора.

Применительно к ПИ закон Фиттса (Fitts's Law) гласит, что «время, требуемое для позиционирования на какой-либо элемент, есть функция от расстояния до этого элемента и от его размера». То есть, чем больше объект и чем ближе он к текущему положению мыши, тем быстрее человек сможет на него щелкнуть.

Закон Хикаутверждает, что время реакции при выборе из некоторого числа альтернативных сигналов зависит от их числа.

$$T = a + b \log_2(n + 1)$$

где T – общее время реакции;

a и b – константы, которые описывают индивидуальные особенности когнитивной сферы (например, такие как задержка перед выполнением задания и индивидуальный коэффициент скорости принятия решения);

n – количество равнозначных альтернативных вариантов, из которых осуществляется выбор.

В соответствии с законом Хика, чем больше пунктов в выборе, тем дольше человекбудет думать, что выбрать. Поэтому рекомендуется делать не больше 5-7 пунктов в меню, использовать группировки, выделения и т.п.

При проектировании интерфейсов закон Хика помогает определить оптимальное количество объектов в однородном массиве (например, в меню), а закон Фиттса –оптимальный с позиции скорости реакции размер элемента.

Модель GOMS (*the model of goals, objects, methods and selection rules)* –модель целей, объектов, методов и правил выбора. Модель GOMS позволяет предсказать, сколько времени потребуется опытному пользователю на выполнение конкретной операции при использовании данной модели интерфейса.

Модель GOMS основана на оценке скорости печати. Время, требуемое для выполнения какой-то задачи системой «человек – компьютер», является суммой всех временных интервалов, которые потребовались на выполнение элементарных действий, составляющих данную задачу. Лабораторным путем установлены стандартные средние интервалы для некоторых действий, выполняемых пользователями:

K =0,2 с. – нажатие клавиши;

P = 1,1 с. – указание на какую-то позицию на экране монитора;

H = 0,4 с. – перемещение руки с клавиатур на «мышь» или обратно;

M = 1,35 с. – мыслительная операция, выбор пользователем следующего действия;

R – время ожидания реакции компьютера.

Выполнение расчетов по модели GOMS позволяет сравнивать длительность выполнения одной и той же операции при различных вариантах организации элементов ПИ.