

Южно-Уральский профессиональный институт

Кафедра математики, информатики и вычислительной техники

Курс лекций по дисциплине

«Человеко-машинное взаимодействие»

для специальности 230105.65 Программное

обеспечение вычислительной техники и

автоматизированных систем

Челябинск

2012

Лекция 1.

Человек и компьютерные среды

План

1. Общие понятия ЧМВ
2. Психология пользователей, восприятие и внимание человека
3. Информационные процессы человека: память и познание

1.1. Общие понятия ЧМВ

Человеко-машинное взаимодействие – это дисциплина, объединяющая знания в областях: психологии познания, проектирования программного обеспечения и компьютерных систем, социологии и организации бизнеса, эргономики и системного анализа, управления процессами и промышленного дизайна. Внедрение компьютеров практически во все стороны жизни требует от современного специалиста в области компьютерных технологий умения разработать или адаптировать пользовательский интерфейс под широкий класс пользователей, обеспечить эффективное использование компьютерных систем в разных приложениях.

Совокупность двух систем: вычислительная система(далее ВС) + Человек

ЧМВ как наука обеспечивает нас знаниями о взаимодействии человека и компьютера для того, чтобы система эффективно выполняла поставленные задачи, а человек, как управляющее звено чувствовал себя естественно и удобно.

Пользовательский интерфейс (далее ПИ) – это совокупность информационной модели разрабатываемой системы, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели в процессе работы с приложением. Модель – состав и взаимодействие реальных компонентов; средства и способы; состав аппаратного и программного обеспечения в распоряжении пользователя, определяемые характером решаемой задачи.

Цель создания ПИ в том, что бы отобразить информацию настолько эффективно, на сколько возможно для человеческого восприятия и структурировать отображение на дисплей таким образом, что бы привлечь внимание к наиболее важным единицам информации.

Качество ПИ является самостоятельной характеристикой ВС, сопоставимой с такими ее показателями, как надежность и эффективность использования вычислительных ресурсов.

Разработчик должен знать что такое хороший интерфейс и как его построить. Главное обратить внимание пользователя на изменения в системе и своевременно реагировать на них. В этом случае эффективность – это необходимое время реакции пользователя (время ответа).

Качество интерфейса сложно оценить количественно из-за разнородности его показателей. Однако набор следующих показателей поможет получить его объективную оценку:

1. Время для достижения заданного уровня знаний и навыков по работе с приложением (время освоения).
2. Сохранение полученных навыков по истечению некоторого времени.
3. скорость решения задачи. При этом должна оцениваться не быстрдействие системы и скорость ввода с клавиатуры, а время, необходимое для решаемой задачи (например пользователь должен обработать за час не менее 200 документов с ошибками).
4. субъективная удовлетворенность пользователя при работе с системой по n-бальной шкале.

Основные свойства разработчика ПИ

1. естественность (интуитивность)

Пользователя не следует вынуждать существенно изменять привычные способы решения задач, когда работа с системой не вызывает у пользователя необходимого поиска элементов интерфейса для поставленной задачи. Целесообразно сохранить обозначения и терминологию данной предметной области.

2. согласованность (непротиворечивость)

Обеспечение преемственности знаний и навыков, когда в процессе работы с системой пользователем использовались некоторые приемы работы с некоторой частью системы, то в другой части приемы должны быть идентичны. Согласованный интерфейс является узнаваемым и предсказуемым, а также соответствующим нормам.

3. избыточность

Пользователь должен вводить только минимальная информация для работы или управления системой. Необходимо избавиться от повторного ввода информации.

4. дружелюбность (непосредственный доступ к системе помощи)

Когда в процессе работы система обеспечивает пользователя сообщениями об ошибках и информацией что система делает. Сообщение об ошибке должно быть полезным и понятным пользователю. Эффективный интерфейс должен предотвращать ситуации, когда пользователь может ошибиться, и предотвращать последствия возможных ошибок.

5. гибкость (адаптивность)

Когда интерфейс системы может обслуживать пользователей с различным уровнем подготовки. Предполагается возможность изменения структуры диалога или входных данных. Для неопытных пользователей интерфейс может быть организован, как иерархическая структура меню, а для опытных, как комбинации клавиш.

6. простота интерфейса

Обеспечение легкости в его изучении и использовании кроме того он должен предоставлять доступ ко всему перечню возможностей, предусмотренных приложением (что противоречит простоте).

Эффективный интерфейс призван сбалансировать эти цели. Один из путей – представление на экране информации минимально необходимой для выполнения пользователем очередного шага задания. Другой путь – представление на экране элементов с учетом их смыслового значения и логичной взаимосвязи, чтобы включить ассоциативное мышление пользователя.

Использование иерархического каскадного меню сокращает объем информации.

1.2. Психология пользователей, восприятие и внимание человека

Роль пользователя в интерфейсе очень важна. Проектирование интерфейса должно базироваться на знании опыта и ожиданий пользователя. Разработчик должен хорошо знать основные физические, познавательные возможности, а также способности пользователей к восприятию.

Когнитивная психология объясняет то, как работает наш мозг, как мы мыслим, как мы запоминаем, как мы обучаемся. Это информационно-процессуальная модель человеческого познания — модель, которая показывает нам схожесть человеческого познания с компьютером и то, что единая теория вычислений может быть использована для проведения исследований и проектирования в психологии и компьютерной отрасли. Но все же это разные модели восприятия, и человеческое познание много сложнее простой обработки и сохранения информации.

Информационно-процессуальная модель представляет обучение как развивающий процесс, объединяющий прежние навыки, знания и приобретаемый опыт. Следовательно, она крайне необходима при проектировании программного обеспечения.

Восприятие можно определить как перенесение недавнего прошлого в настоящее, потому что только настоящее рождает чувство. Роберт Бэйли (Robert Bailey)

Пока компьютеры не научатся как следует распознавать и воспроизводить человеческий голос, люди будут получать информацию визуально, через дисплей. Изучение возможностей человеческого восприятия важно даже если используются относительно простые технологии работы с интерфейсом программы. Рассмотрим лишь один пример, как компьютерная система должна учитывать наши способности восприятия.

Когда на экране со скоростью молнии появляется и пропадает сообщение, которое невозможно прочитать, что вы обычно чувствуете? Раздражение, не так ли? Наша система восприятия отводит очень маленькое время для реакции на стимул и перевода глаз в точку появления информации. Также нужно время для прочтения сообщения и осмысления прочитанного. Возможности человеческого восприятия должны учитываться при определении временного периода на показ и удаление надписи с дисплея.

► Восприятие не просто видение чего-либо, это комбинирование информационных потоков, поступающих в мозг от наших органов чувств, зрения, слуха, вкуса, обоняния, осязания, со знаниями, опытом прошлого.

► Восприятие — процесс сравнения новых образцов информации со старыми. Информационный поток, состоящий из опыта и ожиданий пользователя, который осуществляется через взаимосвязь человека и компьютера.

Нас окружает масса информации, наши чувства постоянно поглощают ее, а мозг перерабатывает. Мы даже не всегда осознаем этот процесс. Наше внимание привлекают лишь какие-то изменения в окружении. Случалось ли вам при разговоре с кем-нибудь в помещении, где находилось большое количество людей, услышать свое имя в беседе, ведущейся в другом конце комнаты? Вы были уверены, что поглощены собеседником, однако умудрились в общей гамме уловить знакомое до боли сочетание звуков! Это называется феноменом вечеринки. Данный пример подтверждает, что наша сенсорная система постоянно работает, а получаемая информация обрабатывается автоматически, без участия сознания человека.

Любые внезапные и значимые изменения в нашем окружении привлекают наше внимание. Они могут касаться освещения, звуков, движений, цвета или всех этих явлений в комплексе. Вот почему в компьютере применяются многочисленные звуки звонков и колокольчиков.

1.3. Информационные процессы человека: память и познание

Компьютер не имеет чувств, подобных человеческим (или пока не имеет), но люди не способны на многое из того, что может компьютер. Чтобы заставить компьютер помогать нам и развлекать нас, надо больше узнать о том, как работают память и познавательная система человека.

Хранение информации от органов чувств есть установка буферов памяти, где содержатся результаты автоматической обработки информации, полученной от наших органов чувств (например, «феномен вечеринки»). Человек перерабатывает огромное количество информации, даже не осознавая этого. Буферная память сохраняет информацию (аудио-, визуальную и тактильную), которая может быть достаточно объемной и обладать высоким уровнем детализации.

Сообщение должно оставаться на экране столько времени, сколько это необходимо для того, чтобы пользователь не только прочитал его, но и понял. Человеческая система чувств воспринимает информацию от всего, что находится на дисплее компьютера. Анимация на заднем плане забавна, но если вы работаете с ней в окне, ваш мозг будет выполнять слишком много ненужных операций. Ваша система обработки информации будет занята задним планом окна, а не работой. Это приведет к усталости и напряжению глаз.

Постоянная или повторяющаяся стимуляция действительно утомляет сенсорные механизмы, и они становятся менее восприимчивыми и способными к дифференциации изменений. Это называется привыканием, которое применимо к любой сенсорной информации, в том числе информации на мониторе, а также к изменениям в окружающей обстановке. Все факторы, включая свет, температуру, звуки, движения, изменения цвета, также влияют на человеческое внимание.

Следовательно, все элементы компьютерного интерфейса важны и должны иметь строго определенное назначение.

Краткосрочная память — это вторая ступень обработки информации. Воспринятые и обработанные данные переходят из хранилища в краткосрочную память, которая также берет информацию из долгосрочной памяти. Краткосрочная память имеет наименьшую, если можно так выразиться, пропускную способность во всей системе обработки информации. Буферная память ограничена по емкости приблизительно семью (плюс-минус двумя) пред-метами. Новая информация поступает в краткосрочную память, вытесняя старую. Если информация не востребуется, в памяти данного вида она хранится не более 30 секунд. Краткосрочная память как область, отвечающая за процесс мышления, называется рабочей памятью.

При разработке интерфейса вам необходимо знать ограничения и основные характеристики краткосрочной памяти. Например, если пользователи не могут понять информацию на экране и запросят справку по конкретной теме, не позволяйте окну справочной системы закрыть ту информацию, для которой она вызвана! Подобного рода помощь называется «деструктивной», потому что закрывает тот предмет, на котором пользователю нужно акцентировать внимание. Пользователи обычно прибегают к справочной системе два-три раза, пока полностью воспримут информацию.

Очень раздражает необходимость запоминать информацию при переходе от одного экрана на другой, а также перерисовка информации в пределах одного экрана с потерей старой. Компьютер способен одновременно показывать предыдущую и нынешнюю информацию.

Долговременная память есть аналог базы данных телефонной компании или телефонной книги. Такого рода память — хранилище информации с неограниченной емкостью и продолжительностью хранения. Компьютеры также являются достаточно объемными, долговременными хранилищами информации, но имеют свои слабые и сильные стороны. Проблема заключается не в количестве и сроке хранения, а в способе получения доступа к информации.

Слабые и сильные стороны людей и компьютеров

Объект	Сильные стороны	Слабые стороны
Люди	Распознавание образов	Краткосрочная память с малой емкостью
	Переключение внимания	Быстрая потеря данных из краткосрочной памяти
	Бесконечная емкость долговременной памяти	Медленная обработка данных
	Богатая, многокодовая долговременная память	Ошибки

	Способность к обучению	Затрудненный доступ к долговременной памяти
Компьютеры	Память с большой емкостью	Простое сравнение с эталоном
	Долговременная память	Ограниченные способности к обучению
	Высокая скорость обработки	Ограниченная емкость долгосрочной памяти
	Обработка без ошибок	Ограниченная интеграция данных
	Безотказный доступ к памяти	

Проектирование пользовательского интерфейса базируется на знании того, как человек познает и воспринимает. Одна из наиболее важных задач интерфейса: уменьшить доверие пользователя к собственной памяти и использовать преимущества компьютера для поддержки человеческих слабостей.

Вопросы для самоконтроля:

1. Поясните, какие вопросы изучает когнитивная психология и почему следует учитывать психологические аспекты восприятия человека?
2. Определите понятие «восприятие».
3. Какие виды памяти можно выделить для человека?
4. Определите понятие «мнемоника». Приведите примеры использования мнемоник в окружающем вас мире.
5. Приведите сильные и слабые стороны людей в познании и восприятии. Приведите примеры, демонстрирующие сильные и слабые стороны.
6. Приведите сильные и слабые стороны компьютера в познании и восприятии. Приведите примеры, демонстрирующие сильные и слабые стороны.
7. Приведите примеры мнемоник, используемые в интерфейсах программных продуктов.

Лекция 2.

Используемые парадигмы и принципы пользовательских интерфейсов

План

1. Понятие пользовательского интерфейса
2. Популярные стили пользовательского интерфейса
3. Критерии эффективного интерфейса

2.1. Понятие пользовательского интерфейса

Каждую вычислительную систему можно оценивать двумя критериями: точностью и удобством. Для традиционной концепции вычислительной системы *точность* означает, что при поступлении на вход системы заданных значений на ее выходе получаются заданные результаты. Часто начинающие программисты обращают особое внимание на точность результатов, забывая про *удобства* работы с вычислительной системой. Слишком часто разработчики программного обеспечения относятся к операции ввода-вывода как к утомительной последовательности операций чтения и записи, которую нужно пройти, чтобы добраться до «интересной части программы».

Пользователь автоматизированной вычислительной системы имеет право ожидать не только точных этапов обработки, но и удобства в использовании системы. Это значит, что при пользовании вычислительной системой человек не должен существенно менять стиль своей работы. В большинстве случаев сам процесс переработки входных данных в выходные не влияет на это, поскольку пользователь непосредственно этот процесс не чувствует. Обычно у него нет необходимости или желания знать, как совершается эта переработка данных. Реальный механизм передачи управления от рулевого колеса к управляемым колесам автомобиля практически не влияет на водителя, однако для него имеет большое значение форма и расположение рулевого колеса. Аналогично устройство и расположение рабочей станции, формат входных и выходных данных имеют большое значение для пользователя вычислительной системы.

Интерфейс "человек-компьютер" включает все те аспекты автоматизированной вычислительной системы, с которой непосредственно соприкасается пользователь.

Интерфейс — одно из тех слов, над которым мы обычно мало задумываемся. Нельзя сказать, что понятие интерфейса никогда не определялось. В различных источниках (словарях, учебниках и т.п.) понятие интерпретируется разным образом. Приведем несколько таких определений.

► Интерфейс — это «место, где независимая система встречается и взаимодействует или производит коммуникацию с другой такой же». В английском языке это слово используется в двух вариантах: как имя существительное и как глагол.

► Раскрывая термин «интерфейс» узкоспециализировано, можно сказать, что он объединяет устройства ввода и вывода и программное обеспечение, которое обслуживает их. В более широком смысле интерфейс включает в себя все, что помогает пользователю взаимодействовать с компьютером, в том числе документацию, обучение и техническую поддержку.

► Пользовательский интерфейс — это совокупность информационной модели проблемной области, средств и способов взаимодействия пользователя с информационной моделью, а также компонентов, обеспечивающих формирование информационной модели.

► Интерфейс — это способ, которым выполняется какая-либо задача с помощью какого-либо продукта.

► Интерфейс программы — это то, что пользователи видят на экране.

► Интерфейс — это способ, с помощью которого пользователь взаимодействует с компьютером, так же важен, как и вычисления сами по себе. Другими словами, приспособленный к человеческому восприятию интерфейс является фундаментом для построения любого компьютера, операционной системы или программного окружения.

К аппаратному обеспечению компьютерного интерфейса относятся клавиатура, манипулятор типа мыши, джойстика или трекбола, системный блок, монитор. Программное обеспечение пользовательского интерфейса содержит все, что помогает пользователям видеть, слышать, отмечать, трогать на экране компьютера, а также информацию, с которой пользователь работает. Кроме того, в интерфейсе есть печатная и электронная информация — справочники, руководства, учебники и много другой документации, дополняющей программное и аппаратное обеспечение. Это делает взаимодействие с программными и аппаратными средствами интерфейса более удобным и позволяет человеку общаться с компьютером, а компьютеру — представлять информацию пользователю.

Дизайн пользовательского интерфейса должен быть одобрен и принят его пользователями. Без хорошо проработанного интерфейса даже выдающаяся система не будет успешной. Часто пользовательский интерфейс является лишь «одежкой» для программных функций. Норман Кокс (Norm Cox) из Далласа, штат Техас, хорошо известный консультант по проектированию пользовательского интерфейса программ, сказал, что такой подход подобен «наложению губной помады на бульдога». Это относится и к попыткам хорошим интерфейсом скрыть неудачный продукт.

Подобная ситуация возникает, когда вы хотите обновить старый продукт или программу, которая была изначально разработана для серверов. Вероятно, у вас нет желания менять дизайн или способ реализации программы. Вы просто хотите сделать ее более удобной и добавить графику. Существует множество инструментов, позволяющих построить персональный компьютерный пользовательский интерфейс для серверных программ и хранения данных. Обычно конечный продукт не бывает оптимальным и, как говорит Норман Кокс, вы можете получить «злую программу с добрым лицом».

Построение интерфейса для конечного пользователя будет эффективным до тех пор, пока не потребует от пользователя совершить какие-либо действия из-за того, что программа нуждается в дополнительной информации. Наиболее продуктивный путь проектирования оптимального интерфейса — сделать дизайн продукта в соответствии с желаниями, потребностями и опытом пользователя.

Стратегия разработки интерфейса человек-компьютер:

1. Интерфейс человек-компьютер как отдельный компонент системы. Так же как структуры данных в системе можно изолировать от алгоритмов обработки этих структур, мы можем до определенной степени отделить интерфейс человек-компьютер от вычислительной задачи.
2. Возможности аппаратных и программных средств. Разработчики систем, как, естественно, и другие специалисты, пользуются в работе своими старыми навыками. Этот внутренний консерватизм усиливается, а не ослабевает вследствие стремительного развития в последнее время аппаратных и программных средств. Однако невозможно разработать компонент, не понимая возможностей и ограничений основных элементов, из которых он может быть построен.
3. Будьте последовательны. В процессе разработки требуется новизна, причем необходимо следить за тем, чтобы эта новизна не растворилась в мелочах, а также за целесообразностью обилия подходов. В настоящее время многие пользователи имеют доступ к разным системам, и вряд ли они будут менять свои приемы работы при смене систем.
4. Используйте принятые принципы разработки интерфейса. Физическое взаимодействие пользователя с рабочей станцией имеет много общего с взаимодействием человека с машиной вообще. Поэтому существует большое число общепринятых в эргономике рекомендаций, которые легко можно перенести на разработку и организацию рабочей станции. В то же время форма представления информации на экране не одинакова. Графический дизайн зависит от распределения информации на экране, словарного состава предложения, способа выделения ключевых элементов представления данных и т.д. Разработчики должны знать эти принципы.
5. «Поймите» задачу и пользователя. Разработчик должен понимать не только вычислительный процесс, необходимый для решения задачи, но и оценивать действия пользователя, направленные на достижение цели задачи. Ему нужно знать особенности потенциальных пользователей системы.
6. Привлекайте пользователей. Рекомендацию о том, что надо "понимать пользователя и задачу", легче дать, чем выполнить. Вряд ли можно ожидать, что системный аналитик или разработчик хорошо знаком со всеми областями применения своих разработок или глубоко чувствует психологические потребности потенциальных пользователей. Существуют принципы, которым нужно следовать, и которые описаны в любом учебнике по системному анализу. Один из типичных принципов заключается в том, что аналитик получает сведения путем опроса будущих пользователей. Полезный способ подбора нужных вопросов — это

поставить себя на место пользователя, работающего с системой. Однако единственный способ оценки доступности интерфейса — это посмотреть, как на самом деле пользователь взаимодействует с системой в нормальных рабочих условиях. Нужен интерактивный подход, приводящий к разработке опытных образцов интерфейсов, с которыми работают пользователи и которые изменяются в соответствии с их реакцией до тех пор, пока не будет создан приемлемый продукт. Это значит, что нужно применять гибкие методы компоновки элементов интерфейса.

7. Предусматривайте средства адаптации в рамках интерфейса. Хотя общие принципы определяют основу создания интерфейса, они не могут удовлетворять любого пользователя. Разработка интерфейса в расчете на среднего пользователя — это как бы поиск наименьшего общего знаменателя. Точно так же привлечение пользователей к разработке не может гарантировать ее абсолютной приемлемости. Даже если условия задачи остаются практически постоянными, потребности пользователей, как и они сами, меняются. Правильно спроектированный интерфейс должен быть настраиваемым на нужды разных пользователей, а также на одного пользователя в разные периоды его работы.

2.2. Популярные стили пользовательского интерфейса

Развитие программного обеспечения потребовало отдельного изучения и прогресса в области разработки программных Интерфейсов. Их многообразие позволяет выделить ряд стилей пользовательского интерфейса, которые завоевали популярность в индустрии программных средств. Среди них:

- ◆ графический пользовательский интерфейс (GUI);
- ◆ пользовательский Web-интерфейс (WUI);
- ◆ пользовательские интерфейсы карманных устройств (HUI);
- ◆ объектно-ориентированные ПИ. Графический пользовательский интерфейс (Graphical

User Interface — GUI) определяется как стиль взаимодействия «пользователь-компьютер», в котором применяются четыре фундаментальных элемента: окна, пиктограммы, меню и указатели. Иногда GUI-интерфейс называют WIMP-интерфейсом (Windows — окна, Icons — пиктограммы, Menus — меню и Pointers — указатели).

Важнейшие свойства GUI-интерфейса — это возможность непосредственного манипулирования, поддержка мыши или указателя, использование графики и наличие области для функций и данных приложения.

Базовый WUI-стиль (Web User Interface) весьма схож с меню иерархической структуры, которые пользователи знают по опыту работы в средах с неграфическим 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

приложения — по одной за раз — в линейном или нелинейном стиле внутри одного GUI-окна. Во многих отношениях WUI-ориентированные приложения — это «шаг назад в будущее» — или, может быть, нечто худшее, учитывая объемы электронных документов и других материалов в Web.

Приведем основные особенности приложения, использующего WUI-стиль.

- ◆ Информация обычно отображается в единственном GUI-окне, называемом браузером, хотя для представления данных в приложении могут использоваться несколько окон браузеров.

- ◆ Браузер обеспечивает меню для Web-приложения.

- ◆ Клиентская область не содержит традиционных пиктограмм.

- ◆ Поддержка указателя осуществляется в основном для выбора с 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. Действия кнопки 2 мыши также ограничены.

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

Сегодня широко известны два основных класса PDA (Personal Digital Assistant — персональный цифровой ассистент — «карманный» компьютер, предназначенный для выполнения некоторых специальных функций). В одних используется настоящий GUI-стиль как по внешнему виду, так и по поведению, в других применяется подмножество GUI-интерфейса. Для ввода данных пользователем оба класса устройств применяют «жестикуляционный» стиль с пером и сенсорным экраном.

Обычно подобные устройства обладают очень маленьким экраном. Каждая область дисплея PDA меньше, чем большинство окон GUI-ориентированных приложений для на-стольных и портативных систем. Для поддержки PDA обычно используется GUI-ориентированное программное обеспечение для портативных или настольных компьютеров.

Проектирование программных объектов дает возможность предоставить в распоряжение пользователя приложение, обладающее объектно-ориентированным стилем пользовательского интерфейса 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 пользовательский интерфейс 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.

Очевидно, что абсолютное количество пользователей использует интерфейс в стиле мейнфреймов или миникомпьютеров. Однако тенденция к доминированию явно склоняется в пользу GUI-интерфейсов и им подобных. Не совсем ясно, какой стиль пользовательского интерфейса будет доминировать в карманных устройствах, хотя можно возразить, что в любом случае эти стили также представляют собой разновидность GUI-стиля.

GUI-интерфейс преобладает в сфере персональных компьютеров, и количество разновидностей этого стиля невелико. WUI-интерфейсы соответствующих приложений, которые используют GUI-стиль, превалируют в области доступа к сетям Internet и Intranet. Стилиевые детали WUI-интерфейсов мало чем отличаются, подтверждением чему служат диалоговые окна Web-браузеров. Однако в разнообразии деталей ПИ для приложений господствует анархия. Стили ПИ для карманных устройств включают как GUI-стили, так и стили, отличные от GUI.

2.3. Критерии эффективного интерфейса

Основой конкурентоспособности является качество. И хотя кроме качества в конкурентоспособность входят цена, сроки поставки, гарантии, сервисное обслуживание и ряд других слагаемых, именно качеству отдают предпочтение покупатели и заказчики при выборе продукции.

Набор критериев для оценки качества интерфейса и связи между ними можно изобразить в виде схемы (рис. 1).

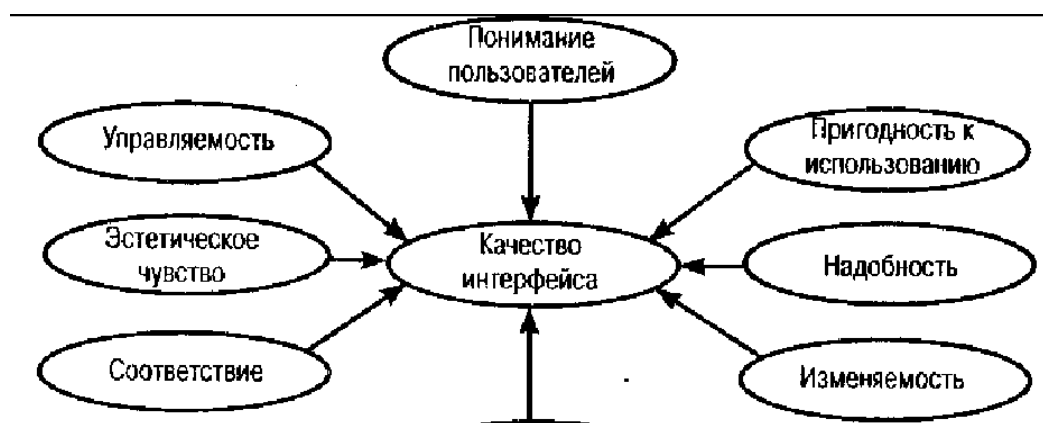


Рис. 1. Критерии качества интерактивного дизайна

Описание критериев интерактивного дизайна:

Критерий проектирования	Описание
Качество интерфейса	В совокупности критерии поднимают ключевой вопрос — как эффективный интерактивный дизайн способствует успеху работы?
понимание пользователей	Насколько хорошо группа проектировщиков понимает потребности, задачи тех людей, для которых предназначен данный продукт? В какой мере отражено данное понимание в программном обеспечении?
Эффективность процесса проектирования	Является ли продукт результатом действительно обдуманного и тщательно реализованного проектирования? Как был профинансирован, распланирован проект? Как решались другие проблемы, например взаимоотношений между членами группы?
Надобность	Что обеспечивает эффективность проекта? Имеет ли продукт общественную, экономическую и другую значимость?
Пригодность к изучению и использованию	Насколько сложен данный продукт в использовании и обучении? Соответствует ли он своему назначению? Все ли особенности продукта наглядно отражены? Как организована его поддержка и имеются ли альтернативные пути достижения поставленных целей в зависимости от опыта работы, навыков и привычек пользователей?
Соответствие	Соответствует ли дизайн продукта решению поставленных проблем? Отвечает ли продукт требованиям практичности и целесообразности? В какой степени решение проблемы соответствует социальным, культурным, экономическим и

	техническим аспектам?
Эстетическое чувство	Насколько использование продукта эстетически приятно? Является ли он цельным с точки зрения дизайна, графики, последовательности действий, информативности? Связаны ли стиль и дух продукта? Отвечает ли дизайн технологическим нормам? Удовлетворяет ли он задаче интеграции программного и аппаратного обеспечения?
Изменяемость	Насколько обоснованна способность продукта изменяться? В какой мере продукт соответствует требованиям индивидуального пользователя и группы? Как проектирование позволяет продукту меняться и подстраиваться под различные, возможно непредвиденные, случаи использования?
Управляемость	Поддерживает ли дизайн продукта понятие «использование» как функциональность или в полном объеме содержания этого термина? В какой мере продукт помогает пользователям управлять такими процессами, как инсталляция, тренировка, сопровождение и др.? Соответствует ли дизайн объекта требованиям коммерческих мероприятий, например конкурсов по использованию продукта, а также концепции «право собственности»?

Основные свойства разработчика ПИ

- естественность (интуитивность)

Пользователя не следует вынуждать существенно изменять привычные способы решения задач, когда работа с системой не вызывает у пользователя необходимого поиска элементов интерфейса для поставленной задачи. Целесообразно сохранить обозначения и терминологию данной предметной области.

- согласованность (непротиворечивость)

Обеспечение преемственности знаний и навыков, когда в процессе работы с системой пользователем использовались некоторые приемы работы с некоторой частью системы, то в другой части приемы должны быть идентичны. Согласованный интерфейс является узнаваемым и предсказуемым, а также соответствующим нормам.

- **неизбыточность**

Пользователь должен вводить только минимальная информация для работы или управления системой. Необходимо избавиться от повторного ввода информации.

дружественность (непосредственный доступ к системе помощи)

Когда в процессе работы система обеспечивает пользователя сообщениями об ошибках и информацией что система делает. Сообщение об ошибке должно быть полезным и понятным пользователю. Эффективный интерфейс должен предотвращать ситуации, когда пользователь может ошибиться, и предотвращать последствие возможных ошибок.

- **гибкость (адаптивность)**

Когда интерфейс системы может обслуживать пользователей с различным уровнем подготовки. Предполагается возможность изменения структуры диалога или входных данных. Для неопытных пользователей интерфейс может быть организован, как иерархическая структура меню, а для опытных, как комбинации клавиш.

- **простота интерфейса**

Обеспечение легкости в его изучении и использовании кроме того он должен предоставлять доступ ко всему перечню возможностей, предусмотренных приложением (что противоречит простоте).

Эффективный интерфейс призван сбалансировать эти цели. Один из путей – представление на экране информации минимально необходимой для выполнения пользователем очередного шага задания. Другой путь – представление на экране элементов с учетом их смыслового значения и логичной взаимосвязи, чтобы включить ассоциативное мышление пользователя.

Использование иерархического каскадного меню сокращает объем информации.

Разработка сценария диалога

Сценарий диалога – это определение всех возможных состояний диалога и путей перехода из одного состояния в другое (за один или несколько шагов). Развитие диалога во времени можно рассматривать, как последовательность переходов системы из одного состояния в другое. Ни одно из них не должно быть «тупиковым», т.е. пользователь должен иметь возможность перейти из любого текущего состояния диалога в требуемое.

Цели разработки сценария: 1) выявление и устранение возможных «тупиковых» ситуация

2) выбор рациональных путей перехода из одного состояния в другое (из текущего в требуемое)

3) выявление не однозначных ситуаций, требующих оказания дополнительной помощи пользователю.

Сложность разработки сценария определяется как функциональными возможностями созданного ПО (т.е. числом и см-ю реализованных функций

обработки) так и степенью неопределенности действий пользователя (которая в свою очередь зависит о выбранной структуры диалога). Наибольшей детерминированностью обладает «меню», наименьшей «вопрос – ответ». Сценарий можно упростить снизив степень неопределенности действий пользователя, используя смещенные структуры диалога (с применением меню) и входной контроль вводимой информации (команд и данных). Сценарий должен также отражать работу средств поддержки пользователя.

Темп ведения диалога

Важность учета темпа ведения диалога была осознана еще в 60-х годах прошлого века, когда появились первые интерактивные системы. Психологи выделили две особенности взаимодействия пользователя с системой.

Первая из них относится к времени ожидания ответа. В обычном разговоре нормальным является время ожидания порядка двух секунд, но если время ожидания превысит четыре секунды, то собеседнику это покажется неестественным, появится желание узнать причину такой задержки. Возникает разрыв в цепи высказывания суждений.

Вторая особенность вытекает из того, что человек не произвольно разбивает свои действия на «группы», выполнение которых может быть легко закончено. Когда такая «группа» действий выполнена, возникает ощущение завершенности (Набор телефонного номера, запись предложения и т.д.). Таким же чувством должны сопрягаться сегменты диалога на рабочем месте. Все дело в том, что в так называемой кратковременной памяти человека может храниться только определенное количество информации (номер телефона, предложение и т.д.). В это время пользователя не стоит отвлекать, поскольку кратковременная память легко теряет информацию. Прерывание же или задержка в достижении завершенности растр, портит настроение. В сложных ситуациях кратковременная память бывает сильно загружена. Психологи считают, что степень сложности проблем, решаемых в уме, зависит от количества и вида информации, которая должна храниться в этой памяти. Кратковременная память рискует потерять свое содержимое (или часть его) под воздействием «шума», рассеянности или невнимательности. Это воздействие резко возрастает, когда человек осознает, что ему приходится ждать неестественно долго. Эффективность умственной деятельности в этом случае резко снижается. Этот спад может рассматриваться, как психологический «скачек вниз».

В зависимости от спецификации ВС и решаемых задач требования к ответу могут быть следующими:

1) более 15 сек. – при единичных простых запросах пользователь может получать ответ тогда, когда ему удобно, т.к. может переключаться на другое дело. В таких случаях диалог исключен.

2) более 4 сек. – при этом сильно затрудняется деятельность по принятию решений и ввода данных. После «главного завершения» пользователь готов ждать до четверти минуты, но потеряет при этом нить решения задачи.

3) 2-4 сек. – для операций, требующих высокой концентрации внимания. Ожидание в 4 сек. для поглощенного делом пользователя, чувствующего моральную ответственность за выполняемую задачу может показаться чрезмерным.

4) меньше 2 сек. – если пользователь должен помнить информацию нескольких последовательных ответов, эти ответы должны быть короткими. При очень сложной работе на АРМ эти 2 сек. являются границей, при переходе которой большинство пользователей испытает один из скачков вниз. По этому диалог должен проектироваться так, чтобы поддержать высокую интенсивность работы пользователя. Для этого желательно (чтобы сохранить тепло человеческой мысли в системе человек-машина) время ответа должно быть меньше одной секунды.

5) почти мгновенный ответ (порядка 0.1 сек.) – это реакция нажатия клавиш, печать символа и т.д. Но слишком короткое время ответа может не удовлетворить пользователя с психологической точки зрения. У оператора замедляется реакция, может даже возникнуть подавленное состояние. Подсознательно он будет чувствовать, что ему навязывают темп работы машины. В таких случаях реализуют специальные задержки, чтобы время ответа было не менее 1,5 сек.

Вопросы для самоконтроля:

1. Можно ли определить понятие «пользовательский интерфейс»?
2. Действительно ли, что понятие интерфейса определено только для программных продуктов? Ответ поясните примерами.
3. Является ли понятие «качество интерфейса» существенным при его разработке и проектировании?
4. Приведите виды моделей, применяемые в разработке и проектировании интерфейсов.
5. Приведите примеры программных продуктов с различными видами пользовательского интерфейса.

Лекция 3.

Среда взаимодействия пользовательских интерфейсов

План

1. Использование цвета в интерфейсе
2. Использование звука и анимации
3. Кнопки

Назначение любого интерфейса — обеспечить коммуникацию между двумя системами, в данном случае людьми и компьютерами. Практические советы и рекомендации по разработке программного обеспечения и пользовательского интерфейса основаны на традиционных способах, используемых при общении людей. Области визуального и графического проектирования базируются на знании теории искусства и визуального восприятия. Эдвард Тафт (Edward Tufte) написал две книги, в которых превосходно описаны наиболее приемлемые способы передачи информации.

Разработчику необходимо осознать важность и сложность графического отображения информации на бумаге и экране компьютера. Каждая строка, каждый управляющий элемент, блок, часть текста, цвет и рисунок, появляющиеся на экране, оказывают влияние на пользователя как по отдельности, так и в комбинации со всем остальным.

3.1. Использование цвета в интерфейсе

Возможно, цвет является наименее используемым элементом в разработке пользовательского интерфейса.

Человек пытается придать значение всему, что видит и воспринимает. Поэтому цвет в пользовательском интерфейсе должен применяться с большой аккуратностью. Часто он используется только в качестве декоративного элемента. Это ограничивает его способность отображать значимую информацию в интерфейсе. Тафт повторяет один из основных принципов использования цвета: «Самое главное — не навреди».

Люди имеют разнообразные физиологические, психологические, культурные и эмоциональные реакции на цвета. Мы говорим: «Она сошла с ума, она в ярости, у нее красная пелена перед глазами», «Он позеленел от зависти», «Ты пожелтел. Ты трус». В нашем сознании мы ассоциируем цвета с эмоциональным состоянием. Артисты, художники и декораторы знают, что теплые цвета (красный, оранжевый и желтый с их оттенками) пробуждают активность и возбуждение, а холодные (синий, фиолетовый, багровый, серый) имеют умиротворяющее и успокаивающее действие.

Использование цвета в пользовательском интерфейсе также может вызывать трудности у людей, имеющих некоторые проблемы со зрением, такие как цветовая слепота. Хотя это и не означает, что люди не различают цветов, данный

тип зрительного дефекта присущ 8% мужского населения и менее 1% женского. Учитывая, что пользователями компьютеров являются люди разного возраста и состояния здоровья, разработчикам приходится учитывать множество нюансов при создании аппаратного обеспечения монитора и разработке программного обеспечения экрана.

Цвет часто применяется для качественного выделения, представления различий в характеристиках, и для количественного выделения, чтобы показать различия в количестве. Присмотритесь внимательнее к странице в местной газете, где приводятся данные о погоде. Цвета используются для показа различных температурных зон на карте. Они варьируются от светлых до темных в соответствии с диапазоном температур, от низких до высоких. Цвета чрезвычайно полезны, поскольку придают дополнительное значение информации. Это свойство используется в различных сферах.

Поскольку цвет является сильным средством привлечения внимания, обилие оттенков заставит пользователей обратить внимание на экран. Это помогает сделать интерфейс более дружелюбным и легким в использовании. Однако данный «эффект Лас-Вегаса» может и отвлекать пользователей при работе. На первый взгляд интерфейсы Macintosh и Windows выглядят теплыми и дружелюбными, поскольку полны ярких цветов. Они привлекают клиентов и пользователей, но когда те часами сидят перед экраном компьютера, им не нужны яркие краски, отвлекающие от работы.

Операционные системы предлагают стандартные цветовые палитры и схемы. Вы должны использовать их при разработке интерфейса. Они были созданы, потому что цвета сложно согласовывать и люди часто ошибаются при выборе цвета или цветовой комбинации.

В пользовательском интерфейсе Windows 3.x, если пользователям не нравятся цвета, используемые по умолчанию, они могут настроить цвета элементов интерфейса на свой вкус. В операционных системах OS/2 и Windows 95 предлагаются дополнительные утилиты — палитры цветовой схемы, где пользователям предоставляются комплекты унифицированных цветовых схем, разработанных специалистами в области графического проектирования.

Пользователи операционной системы OS/2 могут переключать цветовые панели, модифицировать существующие палитры или создавать собственные цветовые схемы. Это невероятно мощная утилита.

Windows 95 также добавил цветовые схемы для «Рабочего стола». В OS/2 даже есть палитра шрифтов, где пользователи могут мгновенно изменить стиль шрифта, размер текста, иконки и названия окон. Это весьма полезно, учитывая различные визуальные способности пользователей, условия работы и типы компьютерных дисплеев.

Рекомендации с точки зрения физиологии

- ◆ Не применяйте одновременно высоконасыщенные и спектрально-интенсивные цвета.
- ◆ Избегайте насыщенно-синего цвета для текстов, тонких линий и малых форм.
- ◆ Не используйте близкие цвета, отличающиеся только количеством синего.

- ◆ Для четкого различения цветов старшим по возрасту операторам назначайте более высокие уровни яркости.
- ◆ Не забывайте о том, что цвета меняют свои оттенки в зависимости от цвета окружения.
- ◆ Помните: величина заметного изменения оттенка варьируется в диапазоне спектра.
 - ◆ Старайтесь выделять края не только цветом.
- ◆ Избегайте красного и зеленого цветов на периферии.
- ◆ Сочетайте противоположные цвета (красный-зеленый или синий-желтый).
- ◆ Для пользователей с дефектами зрения в различении цветов не применяйте близких цветов.

Рекомендации с точки зрения восприятия цветов

- ◆ Не все цвета одинаково хорошо различаемы.
- ◆ Освещенность не равна яркости.
- ◆ Различные оттенки, по существу, имеют различные уровни насыщенности.
- ◆ Освещенность и яркость различимы на отпечатанном документе, но не на цветном дисплее.
 - ◆ Не все цвета одинаково разборчивы и читаемы.
- ◆ Оттенки изменяются при перемене интенсивности и цвета фона.
- ◆ На малых поверхностях цветовые различия плохо заметны.

Рекомендации с точки зрения познавательности

- Не злоупотребляйте цветом.
- Учитывайте, что цвета на экране и печатных копиях различаются.
- Группируйте взаимосвязанные элементы с помощью общего фона.
- Помните: аналогичным значениям соответствуют аналогичные цвета.
- Не забывайте: яркость и насыщенность привлекают внимание.
- Свяжите степень изменения цвета с важностью события.
- Размещайте цвета в соответствии с их спектральным расположением.
- Теплые и холодные цвета должны указывать на уровень действия.

Существуют дополнительные рекомендации по использованию цвета. Арон Маркус (Aaron Marcus) — известный разработчик интерфейсов и консультант в области компьютерной индустрии, имеет образование в сфере информационных и графических разработок. Маркус применяет следующие принципы к использованию цвета: организация, экономия, выделение, взаимодействие, символизм и коммуникация. Его «10 заповедей по использованию цвета» являются хорошим дополнением к рекомендациям Мерча.

- ◆ Используйте от трех до семи цветов.
- ◆ Применяйте цвета, расположенные в центре и на периферии, следующим образом: красный и зеленый — в центре визуального поля; синий — для слайдов, фона и выделения границ экрана.
- ◆ При модификации размеров и изобразительного ряда выбирайте цвета, которые изменяются минимально от кадра к кадру.
- ◆ Не используйте одновременно цвета высокой интенсивности и спектрально-экстремальные.

- ◆ Применяйте соответствующее кодирование для стандартных элементов.
- ◆ Для элементов, принадлежащих к одной группе, назначайте один и тот же цвет.
- ◆ Выбирайте один и тот же цветовой код для тренинга, тестирования, приложений и публикаций.

Цвет в интерфейсе необходимо применять осторожно и экономно, чтобы каждое дополнение было оправдано с точки зрения затрат. Дисплей может показывать от 256 цветов до 1024 или более, однако не увлекайтесь. Основной принцип: разработайте пользовательский интерфейс сначала в черно-белом варианте и позже, при необходимости, добавьте цвет.

3.2. Использование звука и анимации

Звуковой канал недостаточно используется при взаимодействии человека с компьютером. Это становится еще более очевидным, когда мы начинаем осознавать, что в некоторых (часто критических) ситуациях мы визуальны немощны из-за перенасыщения визуальной системы, плохого освещения или в связи с возрастом. Поэтому данная область заслуживает и должна получать больше внимания.

По вопросу использования звука в качестве обратной связи всегда велось множество дискуссий. Если все хорошо продумано, ненавязчиво, можно по желанию звук включать и отключать, то прекрасно. Однако пользователи сразу начинают искать регулировку звука и проверять, издает ли аудиосистема какие-то сигналы, щелчки, нет ли надоедливых голосов, которые будут мешать работе.

Рекомендации по использованию звука, как и цвета, в большей степени основаны на особенностях человеческого восприятия и обучаемости, нежели на программном обеспечении. Дитерэйдж (Daetherage) дает рекомендации, когда стоит применять звуковые и визуальные средства для отображения информации. Несмотря на то, что сформулированы рекомендации более 20 лет, назад они актуальны и сегодня.

Многие современные рабочие среды являются открытыми, т.е. люди не обязательно должны присутствовать в офисах. Когда пользователь окружен другими людьми, звонящими телефонами, работающими компьютерами, то, вероятно, звуковая форма передачи информации малоэффективна. В такой обстановке служащие часто не в состоянии отличить, чей телефон звонит или чей компьютер издает какие-то звуки.

Большинство современного компьютерного программного обеспечения использует звуковую обратную связь, как правило, короткие «бипы», сигнализирующие о допущенной ошибке или неправильно выбранном варианте. Далее такие, на первый взгляд, незначительные звуки могут надоедать. Иногда работники просто смущаются, совершив ошибку, и, естественно, не хотят, чтобы их коллеги узнали об этом. В подобных ситуациях пользователь сразу отключает звуковую систему, как только садится за компьютер.

Анимации, как и звуку, одинаковое внимание уделяют и разработчики, и пользователи. Сегодня вы можете встретить сотни и даже тысячи иконок на CD ROM или в Internet. При перемещении мыши по различным областям экрана ее

указатель изменяет свою форму для отображения типа действия, которое вы можете совершить с данным объектом или в данной области. Существует множество форм, которые может принимать указатель мыши (стрелка, перемещение, размер, ожидание, запрещение и т.д.). Иконки и указатели с анимацией также могут быть полезными.

Под анимацией понимается изменение во времени визуального представления графического элемента. Аналогично звуку главное преимущество анимации заключается именно в развлекательности действия. Курсор с анимацией упрощает его поиск на экране, особенно это важно для небольших компьютерных записных книжек с малым разрешением. Анимация может использоваться для совершенствования визуальной связи между компьютерами и пользователями.

3.3. Кнопки

Кнопкой называется элемент управления, всё взаимодействие пользователя с которым ограничивается одним действием - нажатием. Эта формулировка, кажущаяся бесполезной и примитивной, на самом деле очень важна, поскольку переводит в гордое звание кнопок многие элементы управления, которые как кнопки по большей части не воспринимаются.

Командные кнопки

Нажатие на такую кнопку запускает какое-либо явное действие, поэтому правильнее называть такие кнопки «кнопками прямого действия». С другой стороны, из-за тяжеловесности этого словосочетания им всегда пренебрегают. С точки зрения разработчика ПО для настольных систем, командные кнопки являются чрезвычайно простыми и скучными. Иная ситуация в интернете, где отсутствие операционной системы (откуда приходят элементы управления) и простота создания новых типов кнопок (это чуть ли не единственный элемент, с которым вообще удастся что-либо сделать) привели к тому, что нестандартные кнопки не создает только ленивый. В то же время, этот самый простой элемент управления имеет больше всего тонкостей.

Размеры и поля. Чем больше кнопка, тем легче попасть в нее курсором. Это правило по мере сил всеми соблюдается, во всяком случае, кнопок размером 5 на 5 пикселей уже практически не встретишь. Однако помимо простоты нажатия на кнопку есть другая составляющая проблемы: пользователю должно быть трудно нажать не на ту кнопку. Добиться этого можно либо изменением состояния кнопки при наведении на неё курсором, либо установлением пустого промежутка между кнопками. Первый способ приобрел существенную популярность в интернете, второй в обычном ПО (он, кстати, более эффективен: одно дело, когда пользователь промахивается мимо кнопки и совсем другое - если, промахнувшись, он ещё и ошибочно нажимает на другую кнопку). Ни тот, ни другой способы не обеспечивают стопроцентной надежности, так что при прочих равных использовать стоит оба.

Другой составляющей проблемы размера кнопок в интернете является несоответствие видимой площади кнопки её действующей площади. В пос-

леднее время, кнопки часто реализуют посредством окрашенных ячеек таблицы, в которых размещается текст, являющийся гипертекстовой ссылкой. Проблема заключается в том, что пользователи воспринимают кнопкой всю ячейку, хотя реально «нажимается» лишь малая её часть. Объем. Кнопка должна (или не должна) быть пользователем нажата. Соответственно, пользователю нужно как-то сигнализировать, что кнопка нажимаема. Лучшим способом такой индикации является придание кнопке псевдообъема, т. е. визуальной высоты. С другой стороны, этот объем плох тем, что при его использовании возникает рассогласование между обликами кнопок прямого и непрямого действия. Разумеется, никто не отменял ещё и тот факт, что псевдообъем кнопок, вообще говоря, в существенной степени есть визуальный шум. Еще с одной стороны, зачастую возникает необходимость максимально повышать шансы нажатия пользователем какой-либо отдельной кнопки (например, «О компании»), в этих случаях псевдообъем этой кнопки (при прочих плоских) сильно повышает вероятность нажатия. Состояния. Кнопка должна как-то показывать пользователям свои возможные и текущие состояния. Количество состояний довольно велико, при этом наборы возможных состояний в ПО и в интернете значительно различаются. Например, кнопка в Windows может иметь шесть состояний: нейтральное, нажатое, нейтральное с установленным фокусом ввода, состояние кнопки по умолчанию, кнопка по умолчанию с установленным фокусом ввода и заблокированное состояние. В интернете обычно используют меньший набор состояний: нейтральное, готовое к нажатию (onMouseOver) и активное (в случаях, когда набор кнопок используется для индикации навигации). Нажатое и заблокированное состояние используются очень редко, а «нейтральное с установленным фокусом ввода» старается, как может, создать браузер. Вообще говоря, обычно, чем больше набор состояний, тем лучше. Но главное не это, а отсутствие дублирования состояний: не должно быть разных состояний, выглядящих одинаково. Также очень важно делать заблокированные состояния действительно заблокированными: так, например, в интернете очень часто встречаются кнопки, нажатие на которые открывают ту же самую страницу, т. е. нажатие которых возможно, но бесполезно. Такие кнопки должны не только выглядеть заблокированными (менее яркими и значительными, нежели обычные), но и не нести гипертекстовых ссылок. Текст и пиктограммы. Все руководства по разработке интерфейса с изумительным упорством требуют снабжать командные кнопки названиями, выраженными в виде глаголов в форме инфинитива (Прийти, Увидеть, Победить). Разработчики же интерфейса с не менее изумительным упорством не следуют этому правилу. Аргументов у них два: во-первых, все так делают, значит, это есть стандарт и ему нужно следовать, во-вторых, нет времени придумывать название. Оба аргумента сильны. Действительно, стандарт. Действительно, нет времени. Но есть два контраргумента: во-первых, это не столько стандарт, сколько стандартная ошибка, во-вторых, думать можно и по дороге домой. Если второй контраргумент особых объяснений не требует, то сущность первого полезно объяснить. Кнопка, запускающая действие, недаром называется командной. С

её помощью пользователи отдают системе команды. Команда же в русском языке формируется посредством глагола в повелительном наклонении (никто особо не хочет слишком персонифицировать компьютер и обращаться к нему в наклонении просительном, т. е. Придите, Увидьте, Победите). Помимо этого, у глагольных кнопок есть одно большое достоинство. По ним понятно, какое действие произойдет после нажатия. Это позволяет как-то разграничить диалоговые окна в сознании (поскольку разные диалоговые окна получают разные кнопки). В результате, из-за увеличения степени уникальности фрагментов системы, обучаться системе получается лучше, нежели с кнопками, одинаковыми везде. Более того, вкупе со строкой заголовка окна, глагольные кнопки создают контекст, что очень полезно при возвращении к прерванной работе. Оказывается возможным не рассматривать все диалоговое окно, чтобы узнать, на каком действии задача была прервана - достаточно просто прочесть надпись на кнопке. Таким образом, следует всемерно избегать создания кнопок с ничего не говорящим текстом, поскольку такой текст не сообщает пользователям, что именно произойдет после нажатия кнопки. При этом есть одна тонкость. Существующие интерфейсы заполнены терминаторными кнопками Ок, Отмена (Cancel) и Применить (Apply), что, собственно говоря, и позволяет разработчикам ссылаться на стандарт. Эти кнопки плохи. С первой кнопкой понятно - это не глагол, а значит, кнопка плоха. Кнопка одна и та же во всех диалогах, значит всё ещё хуже. У кнопки ОК, с другой стороны, есть два достоинства. Во-первых, слишком часто оказывается, что ОК является единственным текстом, который можно уместить в кнопку (если во всех отношениях адекватный глагол слишком длинный). Во-вторых, стандартность этой кнопки приводит к почти мгновенному её распознаванию, что позволяет ускорить работу пользователя с системой (с другой стороны, это распознавание может быть неверным, так что появляется риск человеческой ошибки). Вторая, хоть и почти глагол, плоха, поскольку не дает контекста (к тому же отглагольные существительные воспринимаются медленнее, чем соответствующие глаголы). Главный её недостаток, впрочем, заключается в том, что её, как правило, нечем заменить, так что приходится пользоваться ею. Третья, будучи и глагольной, и сравнительно уникальной, имеет другой недостаток: она почти всегда используется неправильно. На ней написано Применить, но на самом деле её значение совсем иное. Разберем это подробнее. Как правило, разработчики создают диалоговое окно, внизу которого располагают три кнопки: Ок, Применить и Отмена (прямо-таки триединство ошибки). Проблемы наступают тогда, когда пользователь делает что-либо в диалоговом окне и начинает думать, какую кнопку ему нужно нажать. Предположим, он всем доволен и нажимает кнопку ОК. Не считая слабо переданного контекста, все довольно хорошо. Все довольно неплохо, если пользователь нажмет кнопку Отмена - его команды просто не будут обработаны системой. А теперь предположим, что пользователь нажал кнопку Применить. Система выполняет команду пользователя и меняет данные. Начинается самое интересное: теперь кнопка ОК не делает ничего (команда-то уже обработана), помимо закрытия окна. Т.

е. эту кнопку в данном состоянии нужно переименовывать в Заккрыть. Более того. Кнопка Отмена после нажатия кнопки Применить тоже начинает врать пользователю: она не отменяет действие, но просто закрывает окно. Таким образом, если делать интерфейс полностью однозначным, получается гадость: последовательность кнопок Ок, Применить и Отмена после нажатия кнопки Применить превращается в последовательность Заккрыть, Применить, Заккрыть. Помимо того, что это просто глупо, это плохо уже и тем, что пользователь оказывается обманут: он-то думает, что если он нажмет кнопку Отмена, его действия в диалоговом окне не будут приняты системой во внимание. В результате, если пользователь нажмет сначала кнопку Применить, а потом кнопку Отмена, он гарантированно совершит ошибку, в которой виновата система. Напротив, если бы вместо кнопки Применить была бы кнопка Предварительный просмотр, все бы работало великолепно. Мало того, что пользователь не путался бы в кнопках, он мог бы избежать многих ошибок, просмотрев результат своих действий перед их окончательным принятием. Но разработчикам реализовывать режим предварительного просмотра тяжело. Гораздо легче вставить кнопку Применить, а то, что пользователям это вредно, их не касается. Таким образом, кнопка Применить оказывается не просто ненужной, но и откровенно вредной. Её можно применять только в палитрах, заменяя ею кнопку ОК, чтобы показывать пользователю, что палитра не исчезнет с экрана после нажатия кнопки. Разумеется, в этом случае с ней должна использоваться кнопка Заккрыть (вместо кнопки Отмена). Во всех остальных случаях кнопка Применить не нужна. Помимо текста, на кнопках можно выводить пиктограммы. Эта возможность редко используется в ПО, но очень широко в интернете. Формально, на таких кнопках пиктограммы не очень хороши из-за того, что они обычно должны передавать пользователям идею действия (т. е. глагол), а действие плохо передается пиктограммами. Конечно, даже и нераспознанная пиктограмма хороша тем, что она визуально отделяет кнопку от кнопки и для опытных пользователей обеспечивает ускорение при поиске нужной кнопки (пользователь может помнить, что ему нужна кнопка с синим пятном на пиктограмме). Так что, судя по всему, пиктограммы хороши для тех кнопок, для которых пиктограммы нарисовать легко, и для тех кнопок, которые нужны особенно часто (при этом качество пиктограммы особого значения не имеет, важно только различие пиктограмм между собой). С другой стороны, единство и согласованность интерфейса требует, чтобы если уж есть пиктограммы, то уж везде; если же это невозможно, лучше вообще изъять пиктограммы из кнопок, поскольку их эффект невелик, а трудозатраты, уходящие на их создание, значительны. Кнопки доступа к меню. Также к группе командных кнопок относится кнопка доступа к меню. Идея проста. Существует много ситуаций, когда раскрывающийся список не помещается в отведенное для него место, поскольку текст в списке слишком велик. Первое, что приходит в голову, это вставить кнопку, нажатие на которую будет вызывать меню. В самой этой мысли нет ничего плохого, но. Во-первых, недостаток кнопки будет проявляться в том, что, поскольку по условиям задачи, текст не будет виден,

значение кнопки будет менее понятным, чем контекстное меню безо всякой кнопки. Формально, для совсем уж неопытных пользователей, кнопка работать будет, но, как только пользователи подрастут, контекстное меню окажется эффективней. Как никак, в кнопку надо попасть курсором, а в меню попадать не надо, достаточно просто нажать правую кнопку мыши. Во-вторых, само использование кнопки в таком исполнении не совсем правильно, поскольку нарушается принцип единообразия: пользователь нажал на кнопку, а действия как такового и нет (не считать же действием появление меню). В интернете это еще проходит, поскольку там кнопки могут и не выглядеть как кнопки, будучи оформлены как ссылки; в этом случае противоречия не возникает. Суммируя, можно смело сказать, что использовать кнопку для инициирования показа меню можно, но стыдно. Не высший класс. Существуют, впрочем, определенные ситуации, когда такие кнопки очень хороши. Для этого только нужно сделать так, чтобы кнопка была одновременно и командной кнопкой, и показывала меню. Для этого нужно сделать две вещи. Во-первых, нужно разделить кнопку на две области, одна из которых запускает действие, а другая открывает меню. Во-вторых, нужно организовать такой контекст, при котором результат нажатия на кнопку всегда будет понятным. Например, это очень хорошо работает с кнопками Вперед и Назад. Другой пример: иногда бывают ситуации, когда действий может выполняться несколько, но чаще всего нужно только одно. В этом случае пользователи очень быстро обучаются этому действию, имея довольно простой доступ к остальным. В таком исполнении кнопки доступа к меню работают замечательно. Осталось сказать немного. Во-первых, на области, вызывающей меню, обязательно должно находиться изображение направленной вниз стрелки. Во-вторых, эта область должна находиться справа на кнопке, чтобы изображение стрелки не мешало воспринимать текст или пиктограмму на кнопке. Чекбоксы и радиокнопки. Первое, что необходимо сказать про чекбоксы и радиокнопки, это то, что они являются кнопками отложенного действия, т. е. их нажатие не должно инициировать какое-либо немедленное действие. С их помощью пользователи вводят параметры, которые скажутся после, когда действие будет запущено иными элементами управления. Нарушать это правило опасно, поскольку это серьезно нарушит сложившуюся ментальную модель пользователей. В этом заключается общность чекбоксов и радиокнопок, теперь поговорим о различиях. Главное различие заключается в том, что группа чекбоксов даёт возможность пользователям выбрать любую комбинацию параметров, радиокнопки же позволяют выбрать только один параметр. Это сближает эти элементы со списками множественного и единственного выбора соответственно (о которых будет подробнее рассказано ниже). Из этого различия проистекают все остальные. Например, в группе не может быть меньше двух радиокнопок (как можно выбрать что-либо одно из чего-либо одного?). Еще одно следствие заключается в том, что у чекбокса есть три состояния (выбранное, не выбранное, смешанное), а у радиокнопки только два, поскольку смешанного состояния у неё быть просто не может (нельзя совместить взаимоисключающие параметры). Но это было знание вообще.

Теперь перейдем к алгоритму его использования. Всякий раз, когда пользователю нужно предоставить выбор между несколькими параметрами, можно использовать либо чекбоксы, либо радиокнопки (или списки, но о них позже). Если параметров больше двух, выбор прост: если параметры можно комбинировать, нужно использовать чекбоксы (например, текст может быть одновременно и жирным и курсивным); если же параметры комбинировать нельзя, нужно использовать радиокнопки (например, текст может быть выровнен или по левому, или по правому краю). Если же параметров всего два и при этом параметры невозможно комбинировать (т. е. либо ДА, либо НЕТ), решение более сложно. Дело в том, что групп⁷ из двух радиокнопок часто можно заменить одним чекбоксом. Предположим, что нужно дать пользователю выбор: показывать в документе линейки или не показывать. В этом случае логично поместить в диалоговое окно рамку группировки со словами Показывать линейки, а в эту рамку поместить две радиокнопки: Да и Нет. Понятно, что это решение очень тяжеловесно. Можно сделать проще: убрать рамку группировки и радиокнопки, а на их место поместить всего один чекбокс со словами Показывать линейки. В этом случае все будет хорошо. К сожалению, этот метод работает не всегда. Поскольку в самом чекбоксе написано только то, что произойдет после его включения, но не описано, что произойдет, если его не включить, такая конструкция не работает в ситуациях, когда пользователям по той или иной причине функциональность непоставленного чекбокса может быть непонятна. Например, если нужно спросить пользователя, в какой кодировке посылать ему письма, не получится заменить две радиокнопки Windows 1251 и KOI-8 единым чекбоксом KOI-8. Пользователь не обязан понимать, в какой кодировке система будет посылать ему письма по умолчанию. К счастью, такие ситуации редки. Внешний вид. Традиционно сложилось так, что чекбоксы выглядят как квадраты, а радиокнопки - как кружки. Нарушать это правило нельзя. Желательно вертикально располагать чекбоксы и радиокнопки в группе, поскольку это облегчает поиск конкретного элемента. Текст подписей. Каждая подпись должна однозначно показывать эффект от выбора соответствующего элемента. Поскольку радиокнопки и чекбоксы не вызывают немедленного действия, формулировать подписи к ним лучше всего в форме существительных, хотя возможно использование глаголов (если изменяется не свойство данных, а запускается какое-либо действие). Подписи к стоящим параллельно кнопкам лучше стараться делать примерно одинаковой длины. Все подписи обязаны быть позитивными (т. е. не содержать отрицания). Повторять одни и те же слова, меняя только окончания подписей (например, «Показывать пробелы» и «Показывать табуляции»), в нескольких кнопках нельзя, в таких случаях лучше перенести повторяющееся слово в рамку группировки. Если подпись не помещается в одну строку, выравнивайте индикатор кнопки (кружок или квадрат) по первой строке подписи. Взаимодействие. Это может показаться невероятным, но до сих пор в интернете 99% чекбоксов и радиокнопок реализованы неправильно. Дело в том, что создатели языка HTML, ничего не понимавшие в проектировании интерфейсов, были поначалу искренно уверены

в том, что в этих элементах управления нажимается только визуальный индикатор переключения, т. е. кружок или квадратик. На самом деле это совершенно не так! Нажимабельной должна быть ещё и подпись. Но в интернете всего этого нет, поскольку в HTML конструкция чекбоксов и радиокнопок просто не позволяла делать нажимабельными подписи. Сейчас это стало технически возможным (через тег Label), но по инерции и вполне понятной лени никто чекбоксы нормальными не делает. Увы. Другой аспект: при необходимости заблокировать элемент, желательно визуально ослаблять не только квадрат или круг, но и подпись. Вариант для панелей инструментов. Как чекбоксы, так и радиокнопки, бывают двух видов: описанные выше стандартные, и предназначенные для размещения на панелях инструментов. У них есть определенный недостаток: они не различаются внешне (насколько я знаю, ни в одной ОС метода визуального различения не выработано). Это не очень критично, поскольку панелями инструментов пользуются в основном сравнительно опытные пользователи, так что страдать по этому поводу не стоит. Тем не менее, на панелях инструментов полезно располагать группы радиокнопок отдельно от групп чекбоксов (чтобы они не смешивались в сознании пользователей). Вообще говоря, графические версии чекбоксов и радиокнопок можно располагать и в диалоговых окнах. Делать это, однако, не рекомендуется, поскольку в окнах они слишком уж похожи на командные кнопки, кроме того, такие кнопки не подразумевают подписей (которые в диалоговых окнах ничего не стоят, принося в то же время явную пользу). Обратите внимание, что на панелях инструментов чекбоксы и радиокнопки могут быть кнопками прямого действия. Списки. Все часто используемые списки функционально являются вариантами чекбоксов и радиокнопок. Скорость доступа к отдельным элементам и наглядность в них принесены в жертву компактности (они экономят экранное пространство, что актуально, если количество элементов велико) и расширяемости (простота загрузки в списки динамически изменяемых элементов делает их очень удобными при разработке интерфейса, поскольку это позволяет не показывать пользователю заведомо неработающие элементы). Списки бывают пролистываемыми и раскрывающимися, причем пролистываемые могут обеспечивать как единственный (аналогично группе радиокнопок), так и множественный выбор (а la чекбокс); раскрывающиеся же работают исключительно как радиокнопки. Но сначала необходимо рассказать об общих свойствах всех списков.

Ширина. Ширина списка как минимум должна быть достаточна для того, чтобы пользователь мог определить различия между элементами. В идеале, конечно, ширина всех элементов должна быть меньше ширины списка, но иногда это невозможно. В таких случаях не стоит добавлять к списку горизонтальную полосу прокрутки, лучше урезать текст элементов. Для этого нужно определить самые важные фрагменты текста (например, для URL это начало и конец строки), после чего все остальное заменить отточием (...). Поскольку нужно максимально ускорить работу пользователей, необходимо сортировать элементы. Идеальным вариантом является

сортировка по типу элементов. Если же элементы однотипны, их необходимо сортировать по алфавиту, причем списки с большим количеством элементов полезно снабжать дополнительными элементами управления, влияющими на сортировку или способ фильтрации элементов. Если можно определить наиболее популярные значения, их можно сразу расположить в начале списка, но при этом придется вставлять в список разделитель, а в систему - обработчик этого разделителя. Пиктограммы. Уже довольно давно в ПО нет технических проблем с выводом в списках пиктограмм отдельных элементов. Однако практически никто этого не делает. Это плохо, ведь пиктограммы обеспечивают существенное повышение субъективной привлекательности интерфейса и сканируются быстрее «голового» текста.

Раскрывающиеся списки

Самым простым вариантом списка является раскрывающийся список. Помимо описанных выше родовых достоинств списков, раскрывающиеся списки обладают одним существенным достоинством. Оно заключается в том, что малая высота списка позволяет с большой легкостью визуальнo отображать команды, собираемые из составляющих. Раскрывающийся список, как правило, вызывает две проблемы, одна появляется преимущественно в ПО, другая - в интернете. Первая проблема заключается в том, что иногда отсутствие места на экране не позволяет использовать ни чекбоксы с радиокнопками, ни пролистываемые списки множественного выбора. Приходится делать раскрывающийся список, в котором помимо собственно элементов есть «мета-элемент», включающий все элементы из списка. Этому элементу часто не дают названия, оставляя строку списка пустой, что неправильно, поскольку требует от пользователя слишком глубокого абстрагирования. Такой мета-элемент нужно снабжать названием, например, Все значения или Ничего. В интернете проблема иная. Раскрывающийся список часто используется как навигационное меню. Это изначально неправильно, поскольку содержимое такого меню не видно сразу и уж тем более им трудно индцировать пользователям, в каком разделе сайта они находятся. Это, впрочем, не главное. Большая проблема заключается в том, что список снабжают скриптом, который запускается сразу по выбору значения. Такой метод имеет два недостатка. Во-первых, список исторически не является элементом управления прямого действия (как и чекбокс, например), что приводит к потере пользователями чувства контроля над системой. Во-вторых, раскрывающиеся списки довольно сложны, так что пользователи часто совершают моторные ошибки при выборе нужного элемента. Поскольку эта ошибка не может быть обнаружена системой и не всегда обнаруживается пользователями, часты ситуации, когда пользователь (как ему кажется) выбирает один раздел, а перемещается в другой, что совсем нехорошо. Таким образом, навигационные раскрывающиеся списки нужно снабжать кнопкой, которая и будет запускать действие, запускать же действие сразу после выбора элемента в списке нельзя.

Пролистываемые списки

Другим, более сложным вариантом списка является пролистываемый список. Пролистываемые списки могут позволять пользователям совершать как единственный, так и множественный выбор. Одно требование применимо к обоим типам списков, остальные применимы только к одному типу. Размер. По вертикали в список должно помещаться как минимум четыре строки, а лучше восемь. Напротив, список, по высоте больший, нежели высота входящих в него элементов, и соответственно, содержащий пустое место в конце, смотрится неряшливо. Требование выводить полоски прокрутки в больших списках кажется моветоном, но забывать о нем не следует. Списки единственного выбора. Список единственного выбора является промежуточным вариантом между группой радиокнопок и раскрывающимся списком. Он меньше группы радиокнопок с аналогичным числом элементов, но больше раскрывающегося списка. Соответственно, использовать его стоит только в условиях «ленивой экономии» пространства экрана. Списки множественного выбора. С точки зрения дизайна интерфейсов, списки множественного выбора интересны, прежде всего, тем, что их фактически нет в интернете. Технически создать список множественного выбора не проблематично, для этого в HTML есть даже специальный тег. Проблема в том, что такой список в браузере будет выглядеть как список единственного выбора, более того, чтобы выбрать несколько элементов пользователю придется удерживать клавишу Ctrl. Это значит, что воспользоваться таким списком сможет только малая часть аудитории (и даже наличие подсказки у списка положения не исправит). Из-за такой убогой реализации списков браузерами, использовать их, как правило, оказывается невозможно. Приходится использовать чекбоксы. Гораздо лучше обстоят дела в ПО. Возможность безболезненно выводить в списке чекбоксы позволяет пользователям без труда пользоваться списками, а разработчикам - без труда эти списки создавать.

Комбобоксы

Комбобоксами (combo box), называются гибриды списка с полем ввода: пользователь может выбрать существующий элемент, либо ввести свой. Комбобоксы бывают двух видов: раскрывающиеся и расширенные. Оба типа имеют проблемы. У раскрывающегося комбобокса есть проблемы. Во-первых, такие комбобоксы выглядят в точности как раскрывающиеся списки, визуально отличаясь от них только наличием индикатора фокуса ввода (да и то, только тогда, когда элемент выделен). Это значит, что полноценно пользоваться ими могут только сравнительно продвинутые пользователи. В этом нет особой проблемы, поскольку комбобоксом все равно можно пользоваться, как обычным списком. Во-вторых, что гораздо хуже, раскрывающиеся комбобоксы отсутствуют в интернете как класс. Поддержки их нет ни в браузерах, ни в HTML. Проблемы расширенных комбобоксов, напротив, совершенно иные. Их с трудом, но можно реализовать в интернете (через JavaScript). Они имеют уникальный вид, отличающий их от остальных

элементов управления. Зато их сравнительно трудно (хотя и гораздо легче, чем в интернете) реализовать в ПО. При этом расширенный комбобокс потребляет много места на экране.

Поскольку комбобоксы являются гибридами списков и полей ввода, к ним применимы те же требования, что и к их родителям.

Поля ввода

Вместе с командными кнопками, чекбоксами и радиокнопками, поля ввода являются основой любого интерфейса. В результате требований к ним довольно много.

Размеры. Основная часть требований к полям ввода касается размера. Понятно, что размер по вертикали должен быть производным от размера вводимого текста - если текста много, нужно добавить несколько строк (нарушением этого правила регулярно грешат форумы, заставляющие пользователей вводить сообщения в поля ввода размером с ноготь).

С размерами по горизонтали интереснее. Конечно, ширина поля должна соответствовать объему вводимого текста, поскольку гораздо удобнее вводить текст, который видишь. Менее очевидным является другое соображение: ширина поля ввода не должна быть больше объема вводимого в поле текста, поскольку частично заполненное поле выглядит как минимум неряшливо. Отдельной проблемой является ограничение вводимого текста. С одной стороны, ограничение хорошо для базы данных. С другой стороны, всегда найдутся пользователи, для которых поле ввода с ограничением вводимых символов окажется слишком маленьким. Поэтому этот вопрос нужно решать применительно к конкретной ситуации. Если же суммировать информацию из двух предыдущих абзацев, можно определить самую большую ошибку, которую разработчики допускают при создании полей ввода. Всякий раз, когда ширина поля ввода больше максимального объема вводимого в него текста, и при этом объем вводимого текста ограничен, пользователи неприятно изумляются, обнаружив, что они не могут ввести текст, хотя место под него на экране имеется. Соответственно, вообще нельзя делать поле ввода шире максимального объема вводимого в него текста. **Подписи.** Вопрос «где надо размещать подписи к полям ввода?» является одним из самых популярных среди программистов: битвы сторонников разных подходов, хоть и бескровны, но значительны. Аргументов и подходов тут множество, моё личное мнение заключается в том, что, поскольку восприятие подписей занимает определенное время, которого жаль, лучше всего действует следующее простое правило: в часто используемых экранах подписи должны быть сверху от поля (чтобы их было легче не читать), в редко же используемых подписи должны быть слева (чтобы всегда восприниматься и тем самым сокращать количество ошибок).

Подписи к полям ввода имеют определенное отличие от других подписей. В полях ввода подписи можно размещать не рядом с элементом, а внутри него, что позволяет экономить пространство экрана. Подпись при этом выводится в самом поле ввода, точно так же, как и текст, который в него

нужно вводить. Необходимо только отслеживать фокус ввода, чтобы при установке фокуса в поле убирать подпись. Это решение, будучи нестандартным, плохо работает в ПО, но неплохо работает в интернете. Если очень жалко экранное пространство, этим методом стоит пользоваться.

Крутилки

Крутилка (spinner, little arrow) есть поле ввода, не такое универсальное. Крутилки не позволяют вводить текстовые данные, но зато обладающее двумя полезными возможностями. Во-первых, чтобы ввести значение в крутилку, пользователю не обязательно бросать мышь и переносить руку на клавиатуру (в отличие от обычного поля ввода). Поскольку перенос руки с места на место занимает сравнительно большое время (в среднем почти половину секунды), к тому же ещё и сбивает фокус внимания, отсутствие нужды в клавиатуре оказывается большим благом. Во всяком случае, ввод значения в крутилку с клавиатуры достаточно редок, т. е. пользователи воспринимают крутилки целиком и полностью положительно. Во-вторых, при вводе значения мышью система может позволить пользователям вводить только корректные данные, причем, что особенно ценно, в корректном формате. Это резко уменьшает вероятность человеческой ошибки. Таким образом, использование крутилок для ввода любых численных значений более чем оправдано. К сожалению, в интернете нет специального элемента для крутилки. Сделать элемент, похожий на крутилку, можно без труда, создав список множественного выбора высотой в один элемент, но ввод в него с клавиатуры будет невозможен. К счастью, крутилку можно с относительно небольшими затратами сделать в Macromedia Flash.

Ползунки

Как и ранее описанные элементы управления, ползунки позволяют пользователям выбирать значение из списка, не позволяя вводить произвольное значение. Возникает резонный вопрос: зачем нужен ещё один элемент управления, если аналогичных элементов уже полно. Ответ прост: ползунки незаменимы, если пользователям надо дать возможность выбрать значение, стоящее в хорошо ранжирующемся ряду, если: значений в ряду много, нужно передать пользователям ранжируемость значений, необходимо дать возможность пользователям быстро выбрать значение из большого их количества (в таких случаях ползунков оказывается самым эффективным элементом, хотя и опасен возможными человеческими ошибками). Ползунки имеют интересный аспект. Их можно также использовать для выбора текстовых параметров, но только в случаях, когда эти параметры можно понятным образом отранжировать. Случаев таких немало, например, «завтрак», «обед» и «ужин», при отсутствии внешней связи ранжированию поддаются вполне.

Вопросы для самоконтроля:

1. Расскажите о правилах использования цвета при создании программного продукта
2. Что такое анимация и ее особенности и основные характеристики
3. Какое значение имеет звук в программе?

Литература

1. Логунова О.С. Человеко-машинное взаимодействие: теория и практика: Учебное пособие / О.С. Логунова, И.М. Ячиков, Е.А. Ильина. — Ростов н/Д : Феникс, 2006. — 285 с.
2. Гультияев А.К., Машин В.А. Проектирование и дизайн пользовательского интерфейса. — СПб.: Корона принт, 2000. — 352 с.
3. Денинг В., Эсиг Г., Маас С. Диалоговые системы. «Человек-ЭВМ». Адаптация к требованиям пользователя / Пер. с англ. — М.: Мир, 1984.
4. Информационно-управляющие человеко-машинные системы: Исследование, проектирование, испытания: Справочник / Под общ. ред. А.И. Губинского, В.Г. Евграфова. — М.: Машиностроение, 1993.
5. Климов А.П. MS Agent. Графические персонажи для интерфейсов. — СПб.: БХВ-Петербург, 2005. — 352 с.
6. Константин Л. Человеческий фактор в программировании / Пер. с англ. — СПб.: Символ-Плюс, 2004. — 384 с.
7. Коутс Р., Влейминк И. Интерфейс «человек — компьютер»: Пер. с англ. — М.: Мир, 1990.
8. Мандел Т. Разработка пользовательского интерфейса / Пер. с англ. — М.: ДМК Пресс, 2001. — 416 с.
9. Гультияев А. К., Машин В. А. «Проектирование и дизайн пользовательского интерфейса»:- СПб , 2000г.

Лекция 4

Принципы использования пользовательских интерфейсов

План

- 1. Виды адаптации**
- 2. Модификация метода**

1. Виды адаптации

Предварительный анализ (хотя бы и на качественном уровне) возможного сценария диалога позволяет избежать многих проблем на этапе реализации ПО. Если ПО может использоваться группой пользователей с различной степенью подготовки, то в ходе диалога необходимо обеспечить достаточную гибкость. ПО должно адаптироваться (пользователем или автоматически) к любому возможному уровню подготовки пользователя.

Существуют три вида адаптации:

- 1) фиксированная
- 2) полная (автоматическая)
- 3) косметическая

При 1-й адаптации пользователь явно выбирает уровень диалоговой поддержки. Простейший ее вариант – правило двух уровней, когда система обеспечивает два вида диалога. Первый – подробный (для начинающего пользователя), второй – краткий (для подготовленных пользователей). Это правило может быть расширено до n уровней диалога, однако:

- не учитывается, что навыки накапливаются постепенно
- пользователь может хорошо знать одну часть системы и не знать другую
- пользователь сам определяет уровень своей подготовки, что снижает объективность оценки

При полной адаптации система должна сама определять стиль диалога с пользователем или его обучение. В качестве характеристик подготовки пользователя могут быть использованы, например, следующие параметры:

- 1) время на подготовку ответа
- 2) количество обращений за помощью и характер ошибок
- 3) тип запрашиваемой помощи

Однако до сих пор полная автоматизация адаптации практически ни в одной системе не реализована.

Косметическая адаптация не учитывает поведение пользователя, однако позволяет выбрать стиль диалога. Применяются следующие методы:

- 1) использование умолчаний
- 2) использование сокращений
- 3) опережающий ввод ответа
- 4) многоуровневая помощь
- 5) многоязычность

Сущность умолчания состоит в том, что система использует некоторые исходные значения параметра, пока пользователь не изменит его. Тогда, во-первых, начинающий пользователь имеет возможность использовать больше параметров системы по умолчанию. Во-вторых – система может запомнить значения в последнем сеансе работы или наиболее часто используемые. Самый простой способ принятия значения по умолчанию – это нулевой ввод (нажатие клавиши «ввод» в ответ на вопрос системы). При использовании командного языка пользователь просто пропускает параметр, используемый по умолчанию.

При использовании сокращений пользователь вместо полного имени команды может вводить любое допустимое сокращенное обозначение. Но, чтобы, не задумываясь изменить команду корректным сокращением, пользователь должен хорошо представлять набор команд, усвоить «лексику» системы.

2. Модификация метода

Модификацией данного метода является опережающий ввод символов, когда система «узнав» по первым символам команды «дописывает» ее сама (в выдачей на экран), а курсор автоматически перемещается в нужную позицию для ввода параметров этой команды.

При операции ввода ответов пользователь может на очередном шаге диалога вводить не один ответ, а целую их цепочку, упреждая возможные вопросы системы.

На принципе много-уровневой основана работа многих серверов. Help-систем, обучающих гипертекстовых систем. Сначала на экран выводится сообщение начального уровня, а затем пользователь может уточнить полученную информацию, используя переход на более низкий уровень по ключевому слову.

Возможный подход к реализации многоязычности, создание средств реакции системы на действие пользователя (сообщение запроса, подсказка, сообщение об ошибке) отдельно от синтаксиса языка программирования (инструментальных средств).

Пользователь должен работать в соответствии с нормами подн. языка и не зависеть от того, на каком языке разработаны инструментальные средства, которые он использует.

Литература:

1. Логунова О.С. Человеко-машинное взаимодействие: теория и практика: Учебное пособие / О.С. Логунова, И.М. Ячиков, Е.А. Ильина. — Ростов н/Д : Феникс, 2006. — 285 с.
2. Гулъяев А.К., Машин В.А. Проектирование и дизайн пользовательского интерфейса. — СПб.: Корона принт, 2000. — 352 с.
3. Денинг В., Эссиг Г., Маас С. Диалоговые системы. «Человек-ЭВМ». Адаптация к требованиям пользователя / Пер. с англ. — М.: Мир, 1984.

4. Информационно-управляющие человеко-машинные системы: Исследование, проектирование, испытания: Справочник / Под общ. ред. А.И. Губинского, В.Г. Евграфова. — М.: Машиностроение, 1993.
5. *Климов А.П.* MS Agent. Графические персонажи для интерфейсов. — СПб.: БХВ-Петербург, 2005. — 352 с.
6. *Константин Л.* Человеческий фактор в программировании / Пер. с англ. — СПб.: Символ-Плюс, 2004. — 384 с.
7. *Коутс Р., Влейминк И.* Интерфейс «человек — компьютер»: Пер. с англ. — М.: Мир, 1990.
8. *Мандел Т.* Разработка пользовательского интерфейса / Пер. с англ. — М.: ДМК Пресс, 2001. — 416 с.
9. Гулятьев А. К., Машин В. А. «Проектирование и дизайн пользовательского интерфейса»:- СПб , 2000г.

Лекция 5

Процесс проектирования пользовательских интерфейсов

План

1. Выбор структуры диалога
2. Разработка сценария диалога
3. Темп ведения диалога

Необходимо определить:

- 1) структуру диалога
- 2) возможный сценарий развития диалога
- 3) сценарий управляющих сообщений и данных, которыми и общается человек с программой
- 4) визуальные атрибуты отображающие информацию

1. Выбор структуры диалога

Рассмотрим четыре основных варианта.

- диалог типа «вопрос – ответ»
- диалог на основе меню
- диалог на основе экранной формы
- диалог на основе командного языка

Диалог типа «вопрос – ответ»

В каждой точке диалог система выводит в качестве подсказки один вопрос, на который пользователь дает один ответ. В зависимости от полученного ответа система может решить, какой следующий вопрос задавать.

Ответ может вводиться как управляющее сообщение (команды), так и данные. Нет ограничения на тип и диапазон входных данных. Ответы могут даваться на естественном языке, но чаще используются предложения из одного слова с ограниченной грамматикой.

Один из существенных недостатков: отвечать на всю серию вопросов довольно утомительно, хотя человек может уже и знать, какие вопросы задаются системой и как на них отвечать.

К достоинствам можно отнести то, что структура диалога удовлетворяет требованиям различных пользователей и типов данных. Она особенно уместна при реализации диалога с множеством «ответвлений». По-этому этот тип диалога часто используется в экспертных системах.

Диалог на основе меню

Существует несколько основных форматов представления меню на экране:

- список объектов, выбираемых прямым указанием, либо указанием номера
- меню в виде блока данных
- меню в виде строки данных
- меню в виде пиктограмм

Пользователь диалогового меню может выбрать нужный пункт вводя в текстовую строку, либо указывая на нее непосредственно, либо просматривая список и выбирая из него.

Меню – это крайне удобная структура диалога для неподготовленных пользователей. Жесткая очередность открытия и иерархическая вложенность меню могут вызвать раздражение у профессионала, замедлив его работу. Традиционная структура меню не достаточно гибка и не в полной мере согласовывается с методами адаптации диалога (например опережающий ввод).

Диалог на основе экранных форм

В отличие от предыдущих типов диалога экранные формы позволяют вести обработку на одном шаге диалога нескольких (а не одного) ответов. На практике формы используются там, где учет какой-либо деятельности требует ввода достаточно стандартного набора данных.

Человек работает с формой до тех пор, пока не заполнит ее полностью и не передаст системе (например с помощью кнопки «ввод»). Если информация не уместается в одном экране, то данные необходимо разбить на группы, которые отображаются в виде последовательности экранов, при этом при разбиении важно сохранить логические связи. Структура данного диалога обеспечивает высокий уровень поддержки пользователя: для каждого вопроса форма может быть предусмотрено сообщение об ошибках и справочная информация.

С формами могут работать пользователи любой квалификации. По сравнению со структурой типа «вопрос – ответ» данная структура позволяет повысить скорость ввода данных. По сравнению же с меню допускается более широкий диапазон входных данных.

Еще одной областью применения экранной формы является формирование запросов в БД (так называемые запросы по образцу query by example).

Диалог на основе командного языка

Это первая из реализованных структур диалога. Она очень часто используется при загрузке ОС. При такой организации диалога система не выводит ни чего кроме пост-ой подсказки (например на ввод команды), которое означает готовность системы к работе. Каждая команда вводится с новой строки и, обычно, заканчивается нажатием кнопки «ввод». За правильность ввода команд отвечает пользователь. При ошибке система информирует о невозможности выполнения заданной команды, не поясняя, как правило, причины отказа. Данная структура не отличается хорошей поддержкой пользователя и пригодна, в основном, для подготовленных специалистов. Поскольку системе не известно, что намеривается делать пользователь, трудно предлагать реальную помощь в процессе работы, кроме выдачи справок общего характера. В данной структуре управление данными осуществляется с помощью составных командных строк, где ключевое слово для команды (что делать) предшествует списку параметров (входных данных). Параметры задаются либо в позиционной либо в ключевой форме. Недостатком позиционных параметров (особенно, если их много) является то, что вводимые значения должны указываться в строго определенном порядке, нарушение которого плохо диагностируется системой и может повлечь непр-ые последствия. Ключевые параметры уменьшают нагрузку на память, кроме того можно опускать

необязательные параметры, однако пользователь должен знать множество ключевых слов.

К недостатку структуры команд можно отнести то, что она не обеспечивает пользователя поддержкой. Очень сложно использовать все заложенные в ней возможности. Большинство пользователей хорошо знакомо только с весьма ограниченным набором средств, с которыми они работают регулярно.

2.Разработка сценария диалога

Сценарий диалога – это определение всех возможных состояний диалога и путей перехода из одного состояния в другое (за один или несколько шагов). Развитие диалога во времени можно рассматривать, как последовательность переходов системы из одного состояния в другое. Ни одно из них не должно быть «тупиковым», т.е. пользователь должен иметь возможность перейти из любого текущего состояния диалога в требуемое.

Цели разработки сценария: 1)выявление и устранение возможных «тупиковых» ситуация

2)выбор рациональных путей перехода из одного состояния в другое (из текущего в требуемое)

3)выявление не однозначных ситуаций, требующих оказания дополнительной помощи пользователю.

Сложность разработки сценария определяется как функциональными возможностями созданного ПО (т.е. числом и см-ю реализованных функций обработки) так и степенью неопределенности действий пользователя (которая в свою очередь зависит о выбранной структуры диалога). Наибольшей детерминированностью обладает «меню», наименьшей «вопрос – ответ». Сценарий можно упростить снизив степень неопределенности действий пользователя, используя смешенные структуры диалога (с применением меню) и входной контроль вводимой информации (команд и данных). Сценарий должен также отражать работу средств поддержки пользователя.

3.Темп ведения диалога

Важность учета темпа ведения диалога была осознана еще в 60-х годах прошлого века, когда появились первые интерактивные системы. Психологи выделили две особенности взаимодействия пользователя с системой.

Первая из них относится к времени ожидания ответа. В обычном разговоре нормальным является время ожидания порядка двух секунд, но если время ожидания превысит четыре секунды, то собеседнику это покажется неестественным, появится желание узнать причину такой задержки. Возникает разрыв в цепи высказывания суждений.

Вторая особенность вытекает из того, что человек не произвольно разбивает свои действия на «группы», выполнение которых может быть легко закончено. Когда такая «группа» действий выполнена, возникает ощущение завершенности (Набор телефонного номера, запись предложения и т.д.). Таким же чувством

должны сопр-ся сегменты диалога на рабочем месте. Все дело в том, что в так называемой кратковременной памяти человека может храниться только определенное количество информации (номер телефона, предложение и т.д.). В это время пользователя не стоит отвлекать, поскольку кратковременная память легко теряет информацию. Прерывание же или задержка в достижении завершенности растр., портит настроение. В сложных ситуациях кратковременная память бывает сильно загружена. Психологи считают, что степень сложности проблем, решаемых в уме, зависит от количества и вида информации, которая должна храниться в этой памяти. Кратковременная память рискует потерять свое содержимое (или часть его) под воздействием «шума», рассеянности или невнимательности. Это воздействие резко возрастает, когда человек осознает, что ему приходится ждать неестественно долго. Эффективность умственной деятельности в этом случае резко снижается. Этот спад может рассматриваться, как психологический «скачек вниз».

Таким образом, проектирование, размещение данных на экране, предполагает решение следующих задач:

- 1)определение состава информации на экране
- 2)выбор формата представления информации
- 3)определение взаимного расположения данных на экране
- 4)выбор средств привлечения внимания пользователя
- 5)разработка макета размещения данных на экране
- 6)оценка эффективности размещения информации

Общие принципы расположения информации на экране должны обеспечивать пользователю:

- 1)возможность просмотра экрана в логической последовательности
- 2)простоту выбора информации
- 3)возможность идентификации связанных групп информации
- 4)различать исключительные ситуации (сообщение об ошибки или предупреждение)
- 5)возможность определить, какое действие со стороны пользователя требуется (и требуется ли вообще) для продолжения выполнения задания.

При этом следует учитывать физиологические особенности человека, как пользователя интерфейса. Ограниченность кратковременной памяти человека диктует необходимые правила разбиения заданий на этапы, когда на экране одновременно присутствует необходимый объем данных (не более 5-9 символов). Если вся информация исходного документа не помещается на одном экране, некоторые элементы данных могут повторяться на других экранах для сохранения целостности и последовательности обработки. Повторяемая информация не должна менять своего расположения на всех шагах выполнения задания, при этом пользователя не следует заставлять дополнительно обрабатывать информацию на экране. (преобразовывать ее, уточнять по справочникам кодов и т.д.).

Существуют правила рационального размещения данных на экране, регулирующие плотность расположения данных в пределах окна:

- оставлять пустым примерно пол экрана
- оставлять пустой строку после каждой пятой строки таблицы
- оставлять 4-5 пробелов между столбцами таблицы

Фрагменты текста должны располагаться на экране так, чтобы взгляд пользователя сам перемещался в нужном направлении. Содержимое полей не должно прижиматься к краю экрана, а располагаться около его горизонтальной или вертикальной осей. Небольшое меню должно смещаться в левую верхнюю часть экрана, чтобы подчеркнуть симметрию содержимого, и наименование полей одной группы должно выравниваться по вертикали.

У человека наблюдается асимметрия головного мозга. Правое и левое полушарие по разному участвует в восприятии и переработки информации. Например, при запоминании слов ведущую роль играет левое полушарие, а при запоминании образов более активно правое. Информация с правой части экрана поступает в левое полушарие, а с левой – в правое. По этому рекомендуется текстовые сообщения группировать справа, а изображения слева. У женщин асимметрия выражена слабее, чем у мужчин. Учет правой и левой асимметрии памяти имеет существенное значение, если интервалы следования сообщений не превышают 10 сек. По этому эти рекомендации особо актуальны для систем реального времени (СРВ).

Особенности проектирования интерфейса в СРВ

К СРВ предъявляются повышенные требования к надежности и быстродействию систем управления. В таких системах участие оператора проявляется при возникновении внештатных, аварийных и критических ситуациях, но именно такие ситуации вызывают у человека дискомфортное или даже стрессовое состояние. По этому особое значение для СРВ имеет проблема реализации средств поддержки пользователя (по запросу оператора или автоматически, например по истечению некоторого допустимого времени ожидания реакции оператора на возникшую ситуацию).

Очень важно также, чтоб оператор в течение достаточно длительного времени находился в состоянии требуемого уровня готовности. Возникают две взаимосвязанные проблемы: предотвращение как «сенсорного голода», так и чрезмерной сенсорной перегрузки оператора.

Рекомендация по размещению информации на экране, а также характеристики зрения оператора (уже рассмотренные нами) позволяют снизить нагрузку пользователя.

«Сенсорный голод» может возникать при чрезмерном искусственном снижении динамичности отображения текущей ситуации на экране, а также света и звукоизоляции рабочего места (далее РМ). По этому в таких случаях должна быть введена определенная избыточность представленной на экране информации, по сравнению с минимальной необходимой. Например, на экране может появляться сообщение, требующее той или иной реакции оператора, но реально не влияющее на процесс управления. Другой способ снижения «сенсорного голода» основан на использовании мультимедийных технологий, т.е. использование нескольких форм представленной информации. Полимодальная организация представления информации оператору позволяет сразу решить несколько задач:

- 1)удовлетворить «сенсорный голод»

2)при дублировании информации по нескольким каналам сокращается время реакции оператора

3)такой подход позволяет увеличить объем одновременно принимаемой информации

4)привлечение дополнительных сенсорных каналов позволяет разгрузить тот, по которому информация поступает наиболее интенсивно. Но, как всегда в таких случаях, применение мультимедийных технологий значительно усложняет интерфейс. Требуется решение дополнительных задач. Основная из них – согласование информации, поступающей по разным каналам по времени и содержанию, иначе эффект от полимодальности окажется прямо противоположным ожидаемому. Может возникнуть перегрузка оператора и, как следствие, его повышенная утомляемость

Литература

1. Логунова О.С. Человеко-машинное взаимодействие: теория и практика: Учебное пособие / О.С. Логунова, И.М. Ячиков, Е.А. Ильина. — Ростов н/Д : Феникс, 2006. — 285 с.
2. Гульяев А.К., Машин В.А. Проектирование и дизайн пользовательского интерфейса. — СПб.: Корона принт, 2000. — 352 с.
3. Денинг В., Эсиг Г., Маас С. Диалоговые системы. «Человек-ЭВМ». Адаптация к требованиям пользователя / Пер. с англ. — М.: Мир, 1984.
4. Информационно-управляющие человеко-машинные системы: Исследование, проектирование, испытания: Справочник / Под общ. ред. А.И. Губинского, В.Г. Евграфова. — М.: Машиностроение, 1993.
5. Климов А.П. MS Agent. Графические персонажи для интерфейсов. — СПб.: БХВ-Петербург, 2005. — 352 с.
6. Константин Л, Человеческий фактор в программировании / Пер. с англ. — СПб.: Символ-Плюс, 2004. — 384 с.
7. Коутс Р., Влейминк И. Интерфейс «человек — компьютер»: Пер. с англ. — М.: Мир, 1990.
8. Мандел Т. Разработка пользовательского интерфейса / Пер. с англ. — М.: ДМК Пресс, 2001. — 416 с.
9. Гульяев А. К., Машин В. А. «Проектирование и дизайн пользовательского интерфейса»:- СПб , 2000г.

Лекция 6

Модель пользователя

План

1. Модели пользовательского интерфейса

Тысячелетия развития промышленности, прошедшие с момента открытия огня и изобретения колеса до начала XX века, мало повлияли на "соотношения сил"

между синими и белыми воротничками. Например, к 1900г. 95% трудоспособного населения индустриально развитых стран были заняты физическим трудом.

Однако ко времени окончания Второй мировой войны в США уже треть работников обрабатывала информацию, а не материальные объекты, к 1980г. - половина, а в ближайшее время, по некоторым прогнозам, фермеры и рабочие составят лишь 10% американских тружеников. Остальные будут работать с информацией, т.е. сидеть за компьютерами.

Модель пользователя- это совокупность знаний об особенностях работы пользователя с системой, его намерениях, целях и требованиях, которая хранится в памяти интеллектуальной системы. М.П. помогает системе организовать эффективный диалог с пользователем, создает ему психологический комфорт.

Деятельность пользователя персонального компьютера как форма взаимодействия субъекта с миром обусловлена ролью компьютера в решении большого числа задач конкретных профессий и видов деятельности. Эта роль трансформируется и проявляется у пользователей персонального компьютера в виде собственной потребности самореализовываться в этом виде деятельности.

Акмеологическими факторами развития профессионального самосознания пользователей персонального компьютера являются: высокая мотивация деятельности; особенности некоторых личностных качеств (радикализм, нонконформизм, высокий самоконтроль), направленность на достижения, потребность в саморазвитии.

1. Модели пользовательского интерфейса

Ментальная модель.

Модель отражает ожидания человека, работающего с компьютером, и тот опыт, который он получает в результате. Это лишь формальная расшифровка опыта и ожиданий пользователя от окружающего мира.

► Ментальная, или концептуальная, модель — это лишь внутреннее отображение того, как пользователь понимает и взаимодействует с системой. Кэрролл и Олсон (Carroll and Olson) расшифровали ментальную модель как «отображение (в основном) физической системы или компьютерного программного обеспечения, в котором заложена вероятная последовательность действий при выполнении операций ввода и вывода».

IBM констатирует: «Ментальная модель не обязательно точно отображает ситуацию и ее компоненты. Пока ментальная модель помогает людям предсказывать, что произойдет далее, она будет служить основой для понимания, анализа и принятия решений». Люди формируют ментальные модели по ряду причин. Мэхью (Mayhew) выяснила, что они позволяют пользователям:

- ◆ предсказывать (или обозначать невидимые) события;
- ◆ найти причины замеченных событий;
- ◆ определить необходимые действия для осуществления нужных изменений;

использовать их как мнемонические устройства для запоминания событий и связей (отношений);

- ◆ обеспечивать понимание аналогичных устройств;
- ◆ применять стратегии, которые позволяют преодолеть ограничения, заложенные в алгоритме обработки информации.

В основе ментальной модели — все взаимоотношения между пользователями и их компьютерами, поэтому она является фундаментом для выработки принципов и правил пользовательского интерфейса. Почему это так важно, если речь идет о моделях? Мэхью дает хорошее обобщение: «Пользователи всегда имеют ментальные модели и будут разрабатывать и модифицировать их независимо от особенностей системы. Наша задача как проектировщиков пользовательского интерфейса — сделать все возможное, чтобы облегчить процесс разработки эффективной ментальной модели». Когда мы переносим знания об окружающем нас мире в мир компьютеров, начинает действовать концепция метафор.

► Метафора — это «понятие, переносящее свойства или признаки одного объекта на другой для выяснения их сходства или аналогии».

► Метафоры помогают пользователям освоить новые для себя области деятельности (например, работу с текстовым процессором), осмысляя их в терминах области, которая им уже знакома и понятна (например, пишущая машинка). Метафоры помогают проектировщикам, так как использование метафор позволяет им структурировать элементы интерфейса по аналогии с известной пользователям областью.

Метафора «рабочий стол» используется в большинстве современных графических и объектно-ориентированных интерфейсов (ООПИ), компьютерный «Рабочий стол» построен по аналогии с офисным, ведь все пользователи бывали в офисе, знакомы с его оборудованием, знают, для чего предназначены папки, шкафы, телефоны, блокноты. Проектировщики используют эту метафору, чтобы облегчить взаимодействие пользователей с компьютером, да и не только с ним.

При построении интерфейсов рассматривается три вида моделей:

- ◆ концептуальная модель пользователя;
- ◆ модель программиста;
- ◆ модель проектировщика.

Модель пользователя

Программные продукты должны разрабатываться прежде всего для удовлетворения потребностей пользователя. Преимущество применения компьютеров и графических пользовательских интерфейсов бесспорно, но только при условии, что продукт имеет хороший дизайн. Графический пользовательский интерфейс может использоваться для обучения и игры так же хорошо, как и для работы.

Примером использования метафор может служить детская программа «Игровая комната», разработанная Лесли Гримм (Leslie Grimm) и реализованная фирмой Broderbund Software, Inc.

Это очень удачная детская программа, которая была отмечена специальной наградой. Она предлагает: «Добро пожаловать в «Игровую комнату», где дети учатся любить учиться». Графический пользовательский интерфейс прививает навыки работы с компьютером и мышью и, что особенно важно для детей от трех до шести лет, обучает чтению и математике. Программа помогает малышам научиться считать, распознавать буквы и цифры, распознавать звуки и слова, запомнить новые слова, правильно их произносить, вести повествование, слагать и вычитать, планировать и анализировать, пользоваться клавиатурой, развивать творческие способности.

В данном случае интерфейс — это метафора детской комнаты в доме, о которой каждый пользователь имеет свое представление, свою ментальную модель. Интерфейс полон предметов, игрушек, движущихся объектов. Щелчок мышью по некоторым из объектов раскрывает другие «комнаты», где ребенок вовлекается в обучающую игру. Для возврата в «Игровую комнату» служит иконка, расположенная в правом нижнем углу экрана. Это достаточно простой, но эффективный интерфейс: ребенок представляет себе, какие бывают комнаты и как они располагаются в квартире, что позволяет ему легко ориентироваться в программе.

Есть большая разница между взрослыми и детьми. Взрослые не столь охотно по сравнению с детьми исследуют интерфейс. Интерфейс для малышей проектируется так, чтобы объекты и движущиеся предметы заинтересовывали маленьких пользователей. Дети любопытны и щелкают мышкой по всему, что видят на экране.

Как определить вид пользовательской модели? Единственный способ — поговорить с пользователем и посмотреть, как он работает, потому, что ментальная модель базируется на персональном опыте и ожиданиях. GUI рекомендует пять способов сбора информации о пользователях:

- ◆ анализ их задач;
- ◆ интервью с настоящими или потенциальными пользователями;
- ◆ посещения мест их работы;
- ◆ отзывы клиентов;
- ◆ тесты по пригодности.

Даже правильно определив «источник» нужной информации, пользователи склонны рассказывать, что они делают, а не то, что им хотелось бы делать. Такой феномен называется WYKIWYL (аббревиатура от английских слов: What You

Know Is What You Like — «Что Ты Знаешь Есть Что Ты Любишь»). Пользователи — не системные проектировщики и обычно не знакомы с технологией, применяемой в компьютерном программном и аппаратном обеспечении. Они больше полагаются на собственные представления, чем на действительное положение вещей.

Как бы то ни было, вы узнали мнение пользователей продукта. Чтобы иметь представление об общей тенденции, вам необходима информация от достаточного количества пользователей. Помните, нет такого понятия — «рядовой пользователь». Собирайте данные от отдельных людей и групп, которые имеют разные персональные, профессиональные и компьютерные привычки и наклонности.

Модель программиста

Модель программиста — самая легкая для отображения, так как она может быть формально и недвусмысленно описана. На самом деле данная модель есть представленная в определенном виде функциональная спецификация программного продукта.

► Программист — человек, который кодирует программу или систему.

Объекты и данные, составляющие программу, интересны программисту, но не обязательно в плане того, как пользователь взаимодействует с информацией. Например, с точки зрения программиста интерфейсы предназначены для сохранения и восстановления информации, представляют собой поля данных или записи в базе данных. Точка зрения на них у пользователя может быть иной, чем у разработчиков. Одни и те же данные могут быть входом в программу для проверки, личную записную книжку или деловую телефонную книгу.

Программист знает компьютерную платформу, операционную систему, инструменты разработки, руководства и спецификации по программированию, нужные для создания программ. Таким образом, здесь не учитывается умение программиста предоставлять пользователю наиболее приемлемые модели и метафоры пользовательского интерфейса.

Некоторые программисты не знакомы с пользовательской моделью компьютерных систем и программ. У них есть собственные ожидания и опыт работы с компьютером, которые они используют при написании программ. Программисты осведомлены больше о функциях программы, чем об интерфейсе. К примеру, если 24 строки доступны в текстовом окне, программисты могут занять практически все, не думая, насколько пользователю будет удобно это читать.

Модель проектировщика

Работа проектировщика подобна работе архитектора. Как мы уже говорили, создание программного продукта во многом похоже на возведение дома.

Проектировщик (архитектор) узнает идеи, пожелания пользователя (владельца дома), соединяет их со своими на-выками и материалами, необходимыми для программиста (строителя), и проектирует программное обеспечение (дом), которое должно удовлетворять нужды пользователя. На рис. 2

приведена модель проектировщика как нечто среднее между моделью пользователя и моделью программиста. Разработчики обычно не входят в контакт с пользователями, для которых создают программы. Недостающим звеном между пользовательским окружением и программистским миром являются проектировщик пользовательского интерфейса и другие члены команды по разработке.

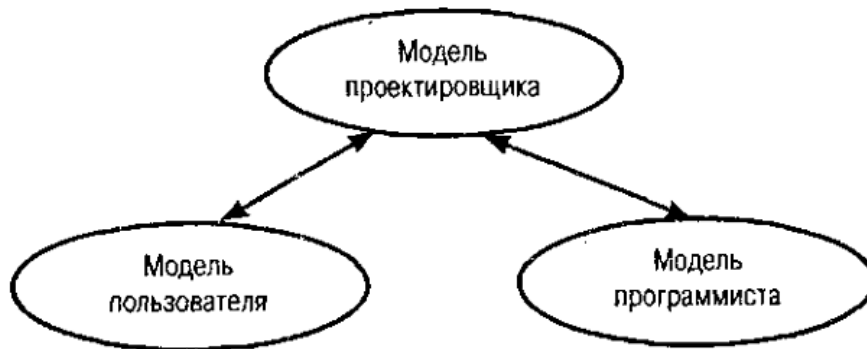


Рис.2. Роль и место модели проектировщика в разработке пользовательского интерфейса

Становится понятным, что ощущения и впечатления от интерфейса есть наиболее заметная часть продукта, которая находится в центре внимания создателей. Отображение информации занимает лишь 10% всего процесса проектирования продукта. Без сомнения, иконки, графика и текст должны иметь приемлемый вид, но это далеко не главная вещь в интерфейсе. Проектирование модели лучше начинать с нижней части «айсберга». На самом деле наиболее важными аспектами являются описание объектов и метафор, их работа. Следовательно, визуальные и эстетические стороны интерфейса, например пиктограммы, должны быть сделаны просто и логично.

Вопросы для самоконтроля:

1. Является ли понятие «качество интерфейса» существенным при его разработке и проектировании?
2. Приведите виды моделей, применяемые в разработке и проектировании интерфейсов.
3. Приведите примеры программных продуктов с различными видами пользовательского интерфейса.
4. Какова роль проектировщика интерфейса?
5. Какова роль программиста интерфейса?

Литература

Основная:

1. Акчурин Э. А. Человеко-машинное взаимодействие. Учебное пособие - М.: СОЛОН - ПРЕСС, 2009.
2. Догадин Н. Б. Архитектура компьютера. Учебное пособие 2-е изд. (эл.) - М.: БИНОМ. Лаборатория знаний, 2012.
3. Магазанник В.Д. Человеко-компьютерное взаимодействие. – М.: Логос, 2007.

Дополнительная:

1. Пятибратов А. П. Вычислительные машины, сети и телекоммуникационные системы. Учебно-методический комплекс - М.: Евразийский открытый институт, 2009.
2. Телекоммуникационные системы и сети: том 1 / Под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005.
3. Телекоммуникационные системы и сети: том 2 / Под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2005.
4. Чекмарев Ю. В. Вычислительные системы, сети и коммуникации - М.: ДМК Пресс, 2009