Лекция 13

Тема. Технологии пользовательского уровня.

План лекции

Выбор пользовательского интерфейса.

Опр. Пользовательский интерфейс – это часть приложения, получающая информацию от пользователя и отображающая ее.

Прежде чем выбрать внешний вид приложения важно подумать, как он будет представлен пользователю. Информацией пользователей обеспечивает пользовательский сервисный уровень. Он отвечает за отображение данных, поступающих от прикладных объектов, а также за отображение объектов данных и получение информации от пользователей.

Пользовательский уровень.

При выборе архитектуры пользовательского уровня ответьте на несколько вопросов.

Ограничивают ли требования к приложению тип пользовательского интерфейса ?

Влияют ли средства защиты (например, брандмауэры) на взаимодействие рабочих станций с серверами?

Какие операционные системы, установленные на рабочих станциях, надо поддерживать?

Какие Web-обозревателя надо поддерживать?

Можно ли на рабочих станциях устанавливать и запускать компоненты COM?

Доступны ли рабочим станциям удаленные COM-компоненты?

Ответы на эти вопросы помогут вам, как разработчику, выбрать наиболее подходящий для приложения тип пользовательского интерфейса.

Требования к приложению

Тип пользовательского интерфейса зачастую зависит от требований к приложению, что облегчает задачу проектной группе. Кроме того, потребности клиентов определяют и платформу, на которой будет реализован презентационный уровень. Например, если пользователи будут обращаться к приложению по Интернету, Web -архитектура — оптимальный выбор. Разработчики могут написать приложение, взаимодействующие с сервером через Интернет, но при этом пользователи должны иметь доступ к приложению с помощью Web-обозревателя.

В пользовательских интерфейсах корпоративных приложений часто много общего, ведь они зависят от принятых на предприятии стандартов. Например, с целью сокращения затрат многие компании применяют стандартный пользовательский интерфейс для всех внутренних разработок. Другие выбрали Web-интерфейс - пользователям много проще путешествовать по Web -страницам, чем работать с обычными приложениями. Кроме того, стоимость сопровождения таких программ, как правило, ниже стоимости сопровождения обычных приложений для среды Windows.

Средства защиты

Средства защиты позволяют:

- аутентифицировать пользователей;
- контролировать доступ к компонентам приложения и данных;
- шифровать информацию.

Брандмауэры и политика защиты влияют на выбор архитектуры приложения. Вопросы защиты касаются в основном Web-приложений, с которыми пользователи работают через Интернет. Однако они имеют отношение и к программам, запускаемым в глобальных вычислительных сетях.

Предлагается четыре метода для разрешения проблем, связанных с брандмауэрами и прокси-серверами.

Перенастройка брандмауэра: брандмауэр можно настроить так, чтобы разрешить запросы к DCOM – компонентам и разрешить прокси-серверам «открывать» IP- адрес пользовательской рабочей станция. Однако обычно такие методы не применяются, так как разработчики не управляют прокси-серверами, а изменение конфигурации брандмауэра чаще всего запрещено корпоративными стандартами.

Туннелирование: DCOM-трафик можно инкапсулировать в HTTP, который, в свою очередь, использует TCP/IP При таком туннелировании обычно применяется порт 80, а так как большинство брандмауэров пропускает трафик этого протокола, применение TCP/IP решит проблему. Однако придется настроить и прокси-серверы разрешив туннелирование для порта 80. Такой способ тоже иногда не годится, так как разработчики не контролируют прокси-сервер.

Использование удаленных сервисов данных (Remote Data Service, RDS): RDS обеспечивает маршалинг объектов ADO Recordset посредством DCOM или HTTP. Вообще говоря, RDS может выполнять маршалинг клиентских вызовов серверных бизнес-объектов, использующих интерфейс IDispatch. При этом применяется COM, что позволяет обходить проблемы брандмауэров и прокси-серверов. Такой метод годятся для любых приложений.

Изменение структуры приложения: приложения можно проектировать так, что они не будут вызывать компоненты COM через брандмауэр или прокси-сервер. Стандартный путь — Web-приложение, реализованное, как правило, посредством активных серверных страниц (Active Serve Pages, ASP), клиентами которого служат Web-обозреватели. ASP-страницы используют сценарии для создания и работы с бизнес-объектами, с помощью которых необходимые Web-страницы создаются и передаются пользователям. В этих страницах разрешены сценарии и компоненты, выполняющиеся на клиенте, но они никогда не обращаются к удаленным COM-объектам, поэтому брандмауэры в данном случае не задействованы.

Ограничения, накладываемые клиентскими операционными системами

Клиентские операционные системы следует учитывать на самой ранней стадии проектирования. Одним из потенциальных ограничений является отсутствие поддержки COM на клиентской платформе. В таком случае приложение не сможет использовать клиентские COM-объекты и устанавливать соединение с удаленными COM-объектами.

Другое ограничение — COM-компоненты распространяются в виде двоичных файлов в формате определенной платформы. Таким образом, при использовании клиентских COM-объектов необходимы отдельная поставка компонентов для каждого типа клиентской платформы. В противном случае придется создавать специальную версию пользовательского уровня, не обращающуюся к этим компонентам.

И последнее ограничение — сервисы пользовательского интерфейса и просмотра Web на разных платформах могут отличаться. Поэтому для работы с несколькими платформами нужно использовать нейтральный каркас пользовательского интерфейса.

Ограничения, накладываемые Web-обозревателями

Возможно, приложению потребуется поддерживать несколько Web-обозревателей даже в среде одной операционной системе. Поэтому при написании Web-приложений важно как можно раньше определить, какие элементы HTML, языка сценариев, объектные модели, компоненты и т. д. поддерживают выбранные обозреватели. Разработчикам придется либо найти общее для всех обозревателей подмножество функций, либо написать пользовательский уровень отдельно для каждого из них.

Разные обозреватели обладают разными возможностями, поэтому выбор поддерживаемых обозревателей поможет вам определять способы реализации функций Web-приложения. Следует изучить различия в объектных моделях разных обозревателей, так как в результате нестыковок будут неправильно работать сценарии. Если вы хотите создать нейтральное (не зависящее от выбора обозревателя) приложение, используйте сценарные языки, совместимые со стандартом European Computer Manufacturers Association (ECMA) — например, Microsoft Jscript.

Важно знать, какие обозреватели популярны у пользователей Если приложение распространяется по корпоративной интрасети, где регламентировано применение определенного обозревателя можно с уверенностью задействовать все его возможности. В противном случае придется учесть в проекте функции других обозревателей.

Например, для компании занимающейся электронной коммерцией, недопустимо отвергать клиента только потому, что у него другой обозреватель. Дабы расширять пользовательскую аудиторию приходится приспосабливаться к старым версиям обозревателей и HTML (такой метод называется добровольной деградацией). Пользователям устаревших обозревателей нужно предоставить возможность просмотра текстовой версии узла или, по крайней мере, сообщить, что им следует обновить свой обозреватель, дав ссылку на узел, с которого можно загрузить его последнюю версию.

COM-компоненты на клиенте

При установке и запуске COM-компонентов «родного приложения на рабочих станциях пользователей проблем, как правило, не возникает — компоненты устанавливаются одновременно с соответствующим Win32-приложением. С Web-приложениями немного сложнее. Если обозреватель поддерживает клиентские COM-компоненты, то они загружаются и автоматически устанавливаются при первом обращении к ним.

Элементы управления Active X

Элементы Active X — это CO M-объекты, загружаемые с Web-сервера и выполняемые на компьютере пользователя с помощью обозревателя. Некоторые пользователи или организации, в которых они работают, считают автоматическую загрузку и установку используемых программ на клиентскую рабочую станцию небезопасной, поэтому запрещают эти действия. Если вы создаете приложение именно для таких заказчиков, ограничьтесь COM-компонентами (в том числе и ActiveX –элементами) выполняющимися на клиенте только при условии, что они уже установлены. Некоторые пользователи самостоятельно принимают решение о загрузке и установке каждого компонента. В этом случае можно использовать клиентские COM-компоненты

Распределенный COM-компоненты.

Чтобы компьютеры пользователей получали доступ к удаленным компонентам, они должны выполнять COM-вызовы с использованием DCOM или RDS. А значит необходимо, чтобы клиентские обозреватели поддерживали создание и сценарные вызовы COM-объектов. Кроме того, для доступа к удаленным компонентам обычно требуется дополнительное ПО или изменение реестра. Если приложение применяет для доступа к COM-компонентам связывание по виртуальной таблице или раннее связывание, на компьютеры пользователей придется установить специальные библиотеки-заместители/представители или библиотеку типов.

Соединения с Интернетом и интрасетью

Если в качестве платформы выбран HTML, пользовательский уровень придется кодировать для нескольких обозревателей. Так как доступ к этому уровню осуществляется с помощью TCP/IP, на решение разработчиков влияют:

- типы пользовательских соединений;
- необходимое число соединений;
- различие скоростей пользовательских соединений.

Выбор архитектуры пользовательского уровня

Выбор интерфейса для сервисов пользовательского уровня иногда представляет собой трудную задачу. Этот раздел поможет вам решить, какой интерфейс — обычный или Web — выбрать, часто приходится реализовать оба варианта

Пользовательский уровень «родных» приложений

Приложения, клиентская часть которых обращается к средствам операционной системы, называют «родными» для этой ОС. Для выполнения своих функций эти приложения используют прикладные интерфейсы операционной системы.

Для разработки «родных» приложений можно применять различные языки программирования и среды разработки. Когда приложение готово, его компилируют, собирают и устанавливают на клиентские системы. Приложения на этих языках являются истинно компилируемыми, поэтому для создания пользовательского интерфейса, состоящего из сложных элементов, многие из которых графические, наилучший выбор — «родное» приложение.

Пользовательский уровень с Web-интерфейсом

Web-интерфейс предлагает почти универсальные методы распространения и готjвые средства отображения. Свободно распространяемые Web –обозреватели обладают простыми интерфейсами, увеличивающими эффективность развертывания приложений. Установка и последующие обновления приложения — самые сложные этапы работы. Web-интерфейс позволяет значительно сократить затраты времени на эти процедуры, так как приложения распространяются среди пользователей не вручную. Поэтому часто предпочтение отдается именно Web -интерфейсу — особенно когда большое значение имеют распространение, развертывание и постоянное сопровождение.

Важно помнить, что Web -интерфейсы изначально предназначались для отображения информации на экране. В последнее же время, благодаря появлению языков сценариев, к их возможностям добавилось и выполнение программ. Перемещение по документам осуществляется посредством гиперссылок, которые способны вызывать выполнение программы на данной странице или в другом файле. Тем не менее, если приложение предназначено для интенсивных вычислений или отображения большого объема информации в графическом виде, Web -интерфейс — не лучшее решение.

Комбинированный пользовательский уровень

Не следует рассматривать «родной» и Web -интерфейсы как взаимоисключающие варианты. Ведь любая программа может обладать двумя интерфейсами. Иногда они похожи, или же один из них отличается какими-то дополнительными функциями. Обратите внимание, что даже выбор двух интерфейсов позволяет повторно использовать значительную часть пользовательских сервисов. Это одно из достоинств пользовательского уровня, созданного для взаимодействия пользовательского интерфейса и бизнес-объектов.

Основы проектирования интерфейса

Чтобы построить прекрасный пользовательский интерфейс, не обязательно быть художником. Большинство принципов проектирования пользовательского интерфейса совпадают с правилами создания художественного произведения. Ведь такие понятия, как композиция, цвет и т. д., одинаковы и для экрана компьютера, и для листа бумаги, и для холста.

Современные средства программирования позволяют создать пользовательский интерфейс, просто перетаскивая элементы управления на форму или Web-страницу. Однако советуем вам не отказываться от предварительного планирования. Сначала лучше прорисуйте пользовательский интерфейс на бумаге, определите необходимые элементы, их относительную важность и взаимосвязи.

Принципы проектирования одинаково подходят как для обычных интерфейсов так и для интерфейсов на основе Web. Программный код, реализующий эти принципы для разных

типов приложений, иногда сильно отличается, однако последовательность проектирования пользовательского интерфейса остается неизменной.

Элементы пользовательского интерфейса

Многие реализации пользовательского интерфейса имеют общие элементы. Их правильное применение повысит эффективность приложения. Хотя внешний вид этих элементов в обычном основанном на Web интерфейсах может быть разным, функционируют они одинаково.

Стили интерфейса

В мире Windows-приложений не все пользовательские интерфейсы выглядят и ведут себя одинаково. Существует три основных стиля и один дополнительный.

Однодокументный интерфейс (Single-Document Interface, SDI): одни из примеров интерфейса такого типа — приложение WordPad из состава Windows. В этой программе можно открыть только один документ. Его надо закрыть прежде, чем. вы сможете открыть новый.

Многодокументный интерфейс (Multiple-Document Interface, MDI): таким интерфейсом обладают, например, Microsoft Word 2000 и Microsoft Excel 2000. В них разрешается одновременно открывать несколько документов, каждый в своем окне.

Интерфейс в стиле Explorer - это окно, состоящее из двух панелей, в одной из которых отображается дерево (слева), а другая представляет собой область отображения текущего элемента дерева (справа). Таков интерфейс Проводника (приложения Explorer) из состава Microsoft Windows.

Отчет: этот стиль обычно считается дополнительным. Информация в таком интерфейсе отображается в любом формате — графическом, табличном, текстовом или комбинированном. Многие приложения позволяют отображать отчеты или печатать их на принтере.

При выборе интерфейса разработчики должны принять во внимание как назначение приложения, так и работу пользователей. Например, для простой программы часы лучше всего выбрать стиль SDI, так как маловероятно, что кому-то потребуется сразу несколько часов одновременно (на крайний случай можно запустить второй экземпляр приложения). Для приложения, обрабатывающего страховки, лучше применить MDI-интерфейс, так как возможна работа сразу с несколькими документами — например, для их сравнения. Интерфейс в стиле Explorer используется во многих новых приложениях, так как позволяет искать и просматривать множество документов, изображений или файлов. Отчеты представляют собой «снимок» состояния информации в определенный момент времени и обычно не предполагают непосредственного взаимодействия пользователя с отображаемой информацией.

Диалоговые окна

Большинство приложений взаимодействуют с пользователем. Для запроса данных, необходимых для работы программы, в Windows-приложениях служат диалоговые окна. Это форма специального типа, которая отображает информацию и, как правило, требует в ответ каких-либо действий со стороны пользователя. Обычно, чтобы продолжить работу с приложением, диалоговое окно нужно закрыть.

Запросы приложения варьируются от простого выбора «да/нет» до выбора из списка с сотнями элементов. Для реализация этих действий служат элементы управления, перечисленные в таблице 1.

Таблица 1 Элементы управления и их назначение

Элемент управления	Описание
Метка (Label)	Только отображает текст
Текстовое поле (text box)	Текст, который пользователь может редактировать. Обычно применяется для ввода информации – например, пароля пользователя

Кнопка (command button)	Кнопки (обычно прямоугольной формы), выполняющие команды пользователя при ее «нажатии»
Флажок (check box)	Набор вариантов, из которых можно выбрать одни или несколько. Показывает состояние какого-либо условия включено/выключено, правда/ложь, да/нет. Если флажки сгруппированы, они независимы друг от друга — пользователь может выбрать любое число вариантов
Переключатель (option button)	Набор вариантов, из которых можно указать только один. Переключатели работают только в группе; выбор одного из них автоматически отключает остальные
Список (list box)	Прокручиваемый список вариантов. Обычно он отображается вертикально в одну колонку, но можно применять списки и из нескольких колонок. Если число элементов списка превышает число отображаемых элементов, должны быть предусмотрены полосы прокрутки. Из списка можно выбрать несколько элементов, удерживая CTRL. Кроме того, список бывает раскрывающимся
Поле со списком (combo box)	Прокручиваемый список, объединенный с текстовым полем. Похож на обычный список, но пользователю разрешается как вводить информацию в текстовом поле, так и выбирать элемент из списка.
Ползунок (slider control)	Позволяет указать значение на шкале. Такими элементами управления обычно задают громкость звука или регулируют цвета изображения
Индикатор (progress indicator)	Показывает, какая часть процесса выполнена к настоящему моменту. Эти элементы управления показывают, что приложение выполняет какую-то работу. Если для этого требуется мною времени, следует предусмотреть вывод времени, необходимого для завершения работы

Если пользователю предоставляется право выбора, желательно, чтобы существ вал вариант по умолчанию.

Композиция

Композиция или размещение элементов пользовательского интерфейса определяет не только его эстетику, но и удобство применения Композиция подразумевает

- размещение элементов управления;
- согласованность элементов управления;
- понятностъ элементов;
- использование разделителей;
- простоту дизайна.

Размещение элемента управления

В большинстве интерфейсов не все элементы равнозначны. Интерфейс нужно проектировать так, чтобы пользователь сразу же понимал, какие элементы важнее других. Поэтому размещение должно подчеркивать иерархию: важные элементы следует помещать на видное место, а маловажные или редко используемые элементы — на менее заметное.
В большинстве языков мира текст читается слева на право и сверху вниз. Это верно не только для книг, но и для экрана компьютера. Поэтому большинство пользователей первым делом обращают внимание на верхний левый угол экрана, следовательно, именно там стоит расположить самые важные элементы интерфейса.
Важно также объединение элементов управления в группы. Группировать элементы следует в соответствии с назначением

Согласованность элементов пользовательского интерфейса

Согласованность элементов — важнейшее качество интерфейса, гарантирующее гармонию. Недостаток согласованности и нелогичность интерфейса вносит в приложение беспорядок, лишают его организованности, что порождает сомнение в его надежности.
Для достижения согласованности следует на ранних фазах проектирования выработать стратегию и соглашения о стиле. Типы элементов управления, их размещение, способы группирования и шрифты надо определять заранее. Для решения этих вопросов можно использовать пробную версию системы или ее прототипы. Итак, какие же вопросы следует согласовать ?
Тип элемента управления: не стоит использовать все доступные элемент управления, хотя, конечно, трудно удержаться от соблазна. Выбирайте только те, которые наилучшим образом подходят для приложения. Несмотря на то, что для представления списочной информации годятся и список, и поле со списком, и табличная форма, и представление в виде дерева, лучше выбрать и везде применять только один из этих элементов.
Свойства: важно согласовать свойства элементов управления. Например, если в одном месте у текстового поля белый фон, то в другом месте не стоит использовать серый фон без веских причин.
Тип формы: для удобства работы с приложением важно согласовать формы — одна с серым фоном и в с трехмерными эффектами, а друга с белым фоном и плоскими элементами — внесут сумятицу и несогласованность.

Понятность элементов

Назначение интуитивно понятных элементов можно определить по их внешнему виду. Пример интуитивно понятного элемента — текстовое поле. Пользователи ожидают, что белое поле, окруженное рамкой, предназначено для редактирования текста. Однако можно отобразить его и без рамки, так что оно будет выглядеть как нередактируемая метка.

Использование разделителей

Разделители подчеркивают важность некоторых элементов, в результате чего работа с приложением становится более удобной. Обычно разделителями служит пустое пространство вокруг или между данными формы. Элементы управления и данные загромождают интерфейс, и становится трудно найти нужные поля и информацию. Разделители же позволяют подчеркнуть важность элементов формы.
Заранее определенные расстояния между элементами и их выравнивание по вертикали и горизонтали делают интерфейс удобнее.

Простота дизайна

Самое важное в дизайне интерфейса — его простота. Если интерфейс приложения сложен, то, скорее всего, и самим приложением трудно пользоваться.
И с эстетической точки зрения ясный и простой дизайн всегда предпочтительней Создание интерфейса на основе существующих методов работы — обычная ошибка многих разработчиков. Например, фирма создает приложение для заполнения страховых документов. Естественно желание создать интерфейс, полностью копирующий бумажные документы. В этом случае возникает несколько проблем:

- форма и размер бумажного документа отличаются от его аналога на экране;

- копирование документа ограничивает возможности разработчиков текстовыми полями и флажками;
- у приложения нет никаких преимуществ по сравнению с бумажными документами.

Гораздо лучше создать новый пользовательский интерфейс, оставив возможность печатать документы, идентичные бумажным. Если применить группировку полей, разбить исходный документ по вкладкам или связанным формам, всю информацию можно будет вводить, не применяя полос прокрутки. Кроме того:

- пользователю придется вводить меньше информации, если спроектированы списки с заранее загруженными вариантами ответа;
- редко применяемые функции можно вынести в отдельную форму;
- пользователю не придется обращаться к каждому элементу интерфейса, если для них будет задано значение по умолчанию (конечно же, обязателен механизм изменения этого значения);
- сложные или редко выполняемые задачи можно упростить с помощью мастера.

Работа с приложением — лучший тест на простоту его интерфейса. Если пользователь не может выполнить какую-либо задачу самостоятельно, вероятно, придется переделать интерфейс.

Цвет и изображение

Использование цвета делает интерфейс значительно более привлекательным, но важно не переусердствовать. Так как большинство современных мониторов способны отображать миллионы цветов, возникает соблазн использовать их как можно больше. Тут-то и появляются проблемы, особенно если все вопросы, касающиеся цвета, не решены на начальных стадиях проектирования интерфейса.

Цветовые предпочтения у всех разные, вкусы пользователей и разработчиков далеко не всегда совпадают. Цвета могут вызывать сильные эмоции, а значения отдельных цветов часто несут конкретную смысловую нагрузку. Поэтому лучше остаться консерватором и применить приятные, нейтральные тона.

Выбор цвета зависит от предполагаемых пользователей приложения и от настроения, которое хотят передать дизайнеры. Ярко-красные, зеленые и желтые цвета годятся для детских программ, но банковские служащие вряд ли оценят их по достоинству.

Яркие цвета в небольших количествах позволяют привлечь внимание к важным элементам интерфейса. Однако их число следует ограничить, а цветовая схема должна соответствовать стилю приложению. По возможности применяйте 16-цветную палитру, так как цвета не включенные в нее, не отображаются на 16-цветных мониторах.

Еще одна проблема — цветовая слепота пользователей. Многие люди не различают комбинации некоторых основных цветов — например, красного я зеленого. Поэтому они не увидят красный текст на зеленом фоне.

Изображения и значки

Картинки и значки добавят привлекательности приложению, но обращаться с ними нужно осторожно. Изображения могут передать информацию вместо текста, но люди относятся к ним по-разному.

Панели инструментов со значками, изображающими различные функции, очень полезны, но если пользователи не смогут сразу же понять, что на них нарисовано, работа только замедлится. При выборе значков для панелей инструментов стоит придерживаться уже действующих стандартов. Другое изображение только закутает пользователя.

При создании значков и изображений лучшее правило — простота. Сложные многоцветные картинки, как правило, плохо отображаются на значке панели инструментов размером 16x16 пикселей.

Шрифты

Так как для передачи информации в основном применяется текст, выбор легко читаемого при разных разрешениях и на разных мониторах шрифта — важная часть проектирования пользовательского интерфейса. Лучше всего задействовать простые шрифты везде, где

возможно. Декоративные шрифты хорошо выглядят на бумаге, но не на экране. К тому же текст, написанный таким шрифтом малого размера, трудно читать.
Обычно в интерфейсе используют стандартные шрифты. В противном случае вместе с приложением придется распространять и применяемый шрифт, так как на компьютерах, где он не установлен, система заменят его на другой, что иногда сильно ухудшает внешний вид приложения.
Общая рекомендация – не использовать более двух шрифтов двух или трех размеров, иначе ваше приложение будет похоже на требование выкупа, составленное из букв, вырезанных из разных газет.

Удобство использования

Насколько удобно приложение, определяют пользователи. Проектирование интерфейса — итерационный процесс. Ведь создать идеальный интерфейс с первой попытки очень сложно. Привлечение пользователей к проектированию на ранних стадиях поможет разработать отличный, удобный интерфейс при меньших усилиях.

Примеры для подражания

Начиная проектировать пользовательский интерфейс, изучите другие приложения, особенно те из них, что хорошо продаются. Для того чтобы сделать их удобными,
затрачено много ресурсов, проведено множество исследований. У интерфейсов таких приложений много общего — панели инструментов, всплывающие подсказки, строки состояния, контекстно-зависимые меню и диалоговые панели с вкладками.
При проектировании стоит учесть и собственный опыт разработчиков как пользователей программного обеспечения. Однако предпочтения небольшой группы разработчиков могут не совпадать со вкусами широкого круга пользователей, поэтому все свои идеи надо проверять на прототипах.
Кроме того, большинство успешных приложений предоставляет пользователю возможность выбрать способ выполнения того или иного действия в зависимости от его предпочтений.

Правила организации пользовательского интерфейса Windows-приложений

Одно из главных достоинств операционных систем Windows состоит в том, что у всех приложений общий интерфейс, Пользователям, научившимся работать с одним из таких приложений, не составит труда изучить другое, но гораздо труднее им привыкнуть к приложению, интерфейс которого им незнаком.
Хороший пример общих элементов интерфейса меню. В большинстве Windows-приложений меню File слева, а меню Help — справа, между ними находятся другие меню, например Edit и Tools. Возможно, кто-то скажет, что название Document лучше, чем File, или что меню Help нужно поместить первым. Ничто не помешает разработчикам отступить от стандартов, но это наверняка вызовет замешательство у пользователей и снизит популярность приложения.
Не менее важно и размещение команд. Например, пользователи привыкли, что команды Copy, Cut иPaste находятся в меню Edit, поэтому они будут недовольны, если вы поместите эти команды в меню File. Лучше всего не отступать от устоявшихся стандартов без особой на то причине.

Доступность функций

Один из важнейших параметров, который проверяется при тестировании удобства использования программы, — доступность различных функций. Если пользователь не может понять, как применить функцию (или не может ее найти), эта функция бесполезна. Чтобы проверить доступность функции, попросите пользователей выполнить определенную задачу, не объясняя, как это сделать. Если им не удастся справится с первой попытки, над доступностью этой функции придется еще поработать.

Модель помощи пользователям

Независимо от того, насколько хорош интерфейс, возникают ситуации, когда пользователю не обойтись без помощи. Поэтому модель помощи пользователям должна включать в себя

как встроенную справочную систему, так и печатную документацию. Кроме того, можно добавить всплывающие подсказки, строки состояния, подсказки «Что это такое?» и мастера.
Модель помощи пользователям, как и другие части приложения, надо проектировать на ранних стадиях процесса разработки. Включенные в нее функции зависят от сложности приложения и от опыта пользователей.

Создание пользовательского интерфейса

Определив вид и тип интерфёйса, разработчики приступают к его реализации.

Реализация пользовательского уровня «родных» приложений

Создание обычных приложений зависит от выбранного языка программирования. Среды разработки Microsoft позволяют создавать интерфейсы на языках Visual Basic, Visual C++ или Visual J++. Такие интерфейсы основаны на элементах управления, различных библиотеках и API операционной системы, входящих в состав библиотек языка программирования.

Реализация Web-интерфейса

Как и все приложения масштаба предприятия, Web-приложения должны динамически отображать информацию из одного или нескольких источников. Для создания таких программ можно использовать ASP из комплекта IIS. В Интернет-приложениях Web-обозреватель реализует пользовательский уровень основе HTML-страниц. Запросы этого уровня передаются Web-серверу по протоколу HTTP. В ответ на запрос клиентского Web-обозревателя на сервере активизируется ASP-страницы, которые способны сгенерировать и вернуть обозревателю обходимую HTML-страницу. ASP можно применять и для создания интерфейса. Следовательно, ASP считается частью пользовательского уровня. Однако в ASP-страницах должны содержаться сценарии, выполняющиеся на сервере и использующие бизнес-объекты среднего уровня. Эти бизнес-объекты способны, в свою очередь, вызывать объекты данных, таким образом получая доступ к уровню данных. С другой стороны, HTML и клиентские сценарии, реализующие пользовательский уровень, реализуются в ASP-страницах. В любом случае, ASP необходим для взаимодействия между пользовательским и прикладным уровнями.

Преимущества ASP

ASP обладает достоинствами, позволяющими избежать некоторых проблем при разработке высококачественных и высокопроизводительных Web-сайтов.
Основное достоинство ASP — привычная модель программирования, похожая на HTML, но дополненная средствами создания сценариев. Сценарии, выполняющиеся на сервере, разрешается писать на любом языке, если на сервере установлен выполняющий его модуль.
Активные страницы выполняются на Web-сервере и полностью контролируют, какие данные будут переданы Web-клиенту. Таким образом защищается Ваша интеллектуальная собственность, будь то информация или код, — на клиентских компьютерах нельзя отследить ни местонахождение данных, ни происхождение HTML страниц. Такой механизм обеспечивает уровень защиты, приемлемый для многих приложений.
ASP устраняет проблемы, связанные с различиями обозревателей. Средствами ASP можно создавать как динамическое, так и статическое наполнение. Первое основано на данных из серверных источников, к которым клиентский компьютер не должен обращаться напрямую Это связано с низкой масштабируемостью клиентских соединений.
С появлением Java-апплетов, элементов Active X, DHTML и ASP стало возможно создавать клиентские Web-приложения, поддерживающие постоянное соединение с сервером.
В ASP включено несколько компонентов автоматизации, выполняющих стандартные действия, например определение характеристик обозревателя, разбор параметров, обмен файлами жетонов и обмен данными между страницами.
ASP — мощный инструмент, позволяющий устранить разрыв между клиентскими функциями представления и серверными бизнес-функциями в трехуровневых приложениях.

Вопросы для проверки.

1. Что такое композиция?
2. Что понимают под удобством интерфейса?
3. Перечислите вопросы, возникающие при проектировании пользовательского уровня.
4. Какими способами можно применять COM-обьекты в пользовательском уровне?
5. Каким образом ASP взаимодействует с пользовательским и прикладным уровнями и уровнем данных ?