Лекция № 13. Объектно-ориентированный подход к проектированию интерфейса.

Концепция интерфейса, управляемого данными.

Разработка, управляемая данными (DCD – Data-centered Design) означает, что проектирование интерфейса поддерживает такую модель взаимодействия пользователя с системой, при которой первичными являются обрабатываемые данные, а не требуемые для этого программные средства. Другими словами, при таком подходе основное внимание пользователя концентрируется на тех данных, с которыми он работает, а не на поиске и загрузке необходимого приложения.

При использовании DCD-технологии основным программным объектом является документ, который представляет собой некоторое абстрактное устройство хранения данных, используемых для выполнения заданий пользователей и для их взаимодействия. Документ должен быть доступен как различным приложениям, используемым для его обработки, так и всем взаимодействующим пользователям.

Основная характеристика ООПИ состоит в том, что они стараются преодолеть существенный недостаток ГПИ – ориентирование на приложения.

Работа с ООПИ основана на прямом манипулировании. У этого метода есть один недостаток – пользователи не видят на экране никаких указаний на то, какие прямые действия они могут совершить с тем или иным объектом. Действия и пункты, осуществляемые с помощью клавиатуры, перечислены на панели меню и в контекстном меню. Следовательно, для работы с ООПИ пользователи должны научиться пользоваться мышью и контекстным меню.

Самым распространенным стилем взаимодействия в ООПИ является последовательность «объект-действие», большинство ГПИ использует стиль взаимодействия типа «действие-объект».

Переход к ООПИ вносит изменения в управляющие элементы. Панель меню ГПИ носит название FEVH (File, Edit, View, Help) – проблемно ориентированная панель. Смысл такой панели определяется моделью «приложение-данные». В ООПИ первый раздел меню File теряет свое значение. Здесь появляется новая структура меню – WOSH (Window, Object, Selected objects, Help).

Объекты представляют собой элементы, которыми можно манипулировать как целыми частями, однако они могут состоять из нескольких других объектов, определенным образом взаимодействующих между собой. На объектах, а не на

приложениях фокусируют пользователи свое внимание при работе с объектно-ориентированным интерфейсом.

Что свидетельствует о разработке объекта, работающего на пользователя? Если вы не в состоянии описать функцию объекта одним коротким предложением, то ваш объект и пользователи будут работать «на разных частотах». Если потребители не смогут без подсказок назвать все важные объекты через пару дней, значит ваш набор объектов неудачен.

Различия между проблемно-ориентированными и объектно-ориентированными пользовательскими интерфейсами.

ПОПИ	ООПИ
Приложение состоит из иконки, первичного окна и вторичных окон	Продукт состоит из набора взаимодействующих объектов или видов объектов
Иконки представляют приложения или открытые окна	Иконки представляют объекты, которыми можно манипулировать напрямую
Пользователи должны запустить приложение, прежде чем приступить к работе с объектами	Пользователи открывают объекты в их представлении на Рабочем столе
Обеспечивает пользователя функциями, необходимыми для выполнения задач	Обеспечивает пользователя запасами, необходимыми для выполнения задач
Акцент делается на основную задачу, по определению приложения	Акцент делается на входные и выходные данные для объектов и задач
Взаимосвязанные задачи поддерживаются другими приложениями	Взаимосвязанные задачи поддерживаются использованием других объектов
Жесткая структура определяется функцией	Гибкая структура определяется объектом
Тренинг сфокусирован на приложении и его функциях	Тренинг фокусируется на общих концепциях, представлениях и функциях
Пользователи следуют структуре приложения	Пользователи могут выполнять задачи по-своему или совершенствовать процесс выполнения
Требуется много приложений – по одному на задачу	Мало объектов – больше повторного использования одних и тех же объектов во многих задачах

Основные характеристики ООПИ по документации IBM.

1. Ориентация на объект.
2. Стандартные объекты и управление.
3. Множественный согласованный просмотр объектов.
4. Композиция объектов и контейнеры.
5. Объединение объектов.
6. Прямое манипулирование.
7. Динамические иконки, отражающие состояние объекта.
8. Мгновенная фиксация изменений.
9. Типы окон, ориентированные на задачу.
10. Пользовательское управление группами окон.
11. Все объекты активны.

Объекты и отношения между ними.

Рассмотренные выше особенности графических интерфейсов, а также положенная в основу их реализации DCD-технология обуславливают необходимость применения для проектирования GUI объектно-ориентированного подхода. Такой подход предполагает использование аналогий между программными объектами и объектами реального мира. С точки зрения пользовательского интерфейса, объектами являются не только файлы или пиктограммы, но и любое устройство для хранения и обработки информации, включая ячейки, параграфы, символы и т.д., а также документы, в которых они находятся.

Объекты, независимо от того, относятся ли они к реальному миру или имеют компьютерное воплощение, обладают определенными характеристиками, которые помогают нам понимать, что они собой представляют, и как они ведут себя в тех или иных ситуациях. Следующие понятия описывают основные аспекты и характеристики объектов, имеющих компьютерное воплощение.

Свойства объектов. Объекты имеют определенные характеристики или атрибуты, называемые свойствами, которые определяют их представление или возможные состояния (например, цвет, размер, дату модификации). Свойства не ограничены внешними или видимыми признаками объекта. Они могут отражать их внутреннюю организацию или текущее состояние объекта.

Операции над объектами. Все действия, которые могут быть выполнены с (или над) объектом, считаются допустимыми операциями. Перемещение или копирование объекта являются примерами операций. Пользователь может выполнять операции над объектами,

используя те или иные механизмы, предоставляемые интерфейсом (командное управление или прямое манипулирование).

Связь (отношения) между объектами. Любой объект тем или иным образом взаимодействует с другими объектами. Во многих случаях взаимоотношения между объектами могут быть описаны как связь определенного типа.

Типы связей между объектами.

Наиболее общими типами отношений являются наборы (Collection), объединения (Constraints), и композиции (Composites).

Набор представляет собой наиболее простой тип отношения, которое отражает наличие у объектов некоторых общих свойств. Результаты запроса (поиска по образцу) или операции множественного выбора объектов – примеры использования данного типа отношения. Важным достоинством этого типа отношения является то, что он позволяет указывать операции, которые должны относиться к определенному набору объектов.

Объединение отражает более тесное отношение между объектами, при котором изменение объекта влияет на некоторый другой объект в наборе. Простейший пример такого отношения – изменение формата соседней страницы при добавлении текста на предыдущей странице.

Композиция имеет место в том случае, когда агрегация нескольких объектов может рассматриваться как новый объект со своим собственным множеством свойств и допустимых операций. Столбец ячеек в таблице и параграф в тексте – это примеры композиций.

Еще один распространенный тип отношений между объектами – контейнер.

Контейнер является объектом, который содержит другие объекты (например, рисунок в документе или документ в папке могут рассматриваться как часть содержимого соответствующего контейнера). Свойства контейнера часто влияют на поведение его содержимого. Это влияние может заключаться в расширении или подавлении некоторых свойств содержащихся в нем объектов или в изменении перечня допустимых операций. Кроме того, контейнер управляет доступом к своему содержимому, а также преобразованием типа (формата) включаемого в него объекта. Это, в частности, может сказаться на результате пересылки объекта из одного контейнера в другой.

Рассмотренные выше аспекты обуславливают необходимость отнесения каждого объекта к тому или иному типу (классу) объектов. Объекты одного типа имеют аналогичные свойства и поведение.

Основные типы объектов интерфейса составляют фундаментальные классы всех объектов, обеспечиваемых операционной системой. Существует три основных типа объектов: объекты-данные, объекты-контейнеры и объекты-устройства.

Многие объекты обладают характеристиками, относящимися более чем к одному классу (пример – папка Входящие: свойства контейнера и устройства). Поэтому необходимо хорошо разбираться в классах объектов интерфейса и их поведении. Объекты должны оправдывать ожидания пользователей в отношении выполняемых ими действий, то есть определять, какие представления могут его отобразить и изменить. Объекты-контейнеры должны обеспечивать представления, соответствующие другим контейнерам, объекты-устройства – предлагать представления, присущие данному устройству и совместимые с другими.

Объекты-данные снабжают пользователей информацией. Они могут представлять любой тип информации, например текст, электронные таблицы, изображения, музыку, записанную речь, видео, анимацию или любую их комбинацию. Поскольку объекты-данные, как правило, ориентированы на продукт, руководства по разработке не дают определения особых объектов данных. Это задача проектировщиков программ.

Объекты-данные также могут быть составными объектами, содержать другие объекты. Такие объекты должны обладать поведением, присущим объектам-контейнерам.

Объекты-контейнеры являются мощным инструментом в руках пользователей для организации их работы. Они могут хранить и группировать любые объекты, в т.ч. и другие контейнеры, представляя их содержимое различными способами, перемещая и копируя объекты с и на контейнеры, а также выстраивая или сортируя содержимое в каком-либо порядке. К типичным контейнерам относятся папки, корзины входящих и исходящих для почты. 3 основных вида контейнеров: рабочее место (Рабочий стол), папки и рабочие области.

Объекты-устройства часто представляют устройства, существующие в реальном мире. Главным назначением объектов-устройств является обеспечение пользователей способами коммуникации и взаимодействия с объектами, связанными с их компьютерами.

Объект-устройство может обладать характеристиками других типов объектов. Например, принтер, факс и корзина для мусора содержат объекты. Принтер имеет очередь вывода на печать, связанную с ним, факс содержит задания и страницы, а корзина для мусора позволяет открывать ее и знакомиться с содержимым. Тип характеристик объекта и поведения любого конкретного объекта будет определять представления, свойственные данному объекту.

Представление объектов.

Определение объектов и представлений наиболее сложная часть процесса разработки пользовательского интерфейса. Окна представления объектов позволяют рассматривать объект и содержащуюся в нем информацию различными способами. При проектировании ООПИ необходимо определить, каким образом пользователи хотели бы работать с объектами, и обеспечить их соответствующими представлениями.

Существует *четыре основных типа представления объектов*: составные, содержание, свойства и система помощи. Представления подаются через окна. Содержимое окна и способы обработки этой информации частично определяются типом представления объекта. Пользователи могут взаимодействовать с объектами, используя прямое манипулирование, а также представления, отраженные в окнах.

Составные представления отражают информацию и объекты, содержащиеся в конкретном продукте, показывая порядок и взаимоотношения с другими компонентами. Такие представления часто являются первичными видами, связанными с объектами данных. Составные представления в значительной степени ориентированы на продукт и задачи, стоящие перед пользователями.

Представление содержания отображает компоненты и содержимое объектов. Такой тип является стандартным для объектов-контейнеров. Порядок представления содержания не обязательно изменяет значения самого объекта при перестановке содержимого.

Любой объект, обладающий свойствами контейнера, может иметь представление содержания. Объекты принтера имеют его в дополнение к представлению свойств принтера. Объекты данных могут также иметь представления содержания, где перечисляются их компоненты, но, поскольку важны отношения между компонентами объекта данных, их порядок роли не играет.

Представления свойств позволяют просматривать и изменять информацию или свойства объектов.

Перечисленными представлениями должны сопровождаться все типы объектов. В проблемно-ориентированных программах опции и свойства, как правило, отражаются в диалоговых окнах внутри приложения. В ООПИ пользователи должны уметь изменять все характеристики объекта в виде свойств (цвета, шрифта, названий и т.д.).

Представления объектов должны быть динамическими и тесно взаимосвязанными. Если пользователь вносит в объект изменения, влияющие на другие представления, то новшества должны отражаться немедленно или как можно быстрее. Использование многочисленных представлений позволяет сразу же увидеть результаты их изменений.

Представление системы помощи. Информация отражается в представлениях системы помощи для поддержки работы пользователя с объектами. С точки зрения разработчика – помощь является видом объекта, однако пользователи, как правило, получают вспомогательную информацию через главное или всплывающее меню. Помощь может оказываться на уровне объекта или конкретного элемента, например для поля ввода (контекстная подсказка).

Помощь пользователям должна оказываться в том виде, в котором она им необходима. Она является представлением объекта и состоит из информации, принимающей любые формы. Необходимо определить, какой тип помощи предпочитают пользователи.

Описание типа объекта (данных, контейнера или устройства) являются техническими терминами, а пользователи должны видеть только названия объектов или классов, например «документ», «папка» или «принтер». Составное представление также является техническим термином, у которого должно быть пользовательское название, например WYSIWYG вид для документа. Представления содержания подаются пользователям как название вида содержания (иконки, древовидная схема, детали и т.п.). Представление свойств является и техническим, и пользовательским термином. Изображается как окно Свойства.

Названия объектов и представлений чаще всего появляются в меню и названиях окон. Каждое открытое окно содержит название объекта и представления на панели названия.

Пользователи должны видеть удобные в применении и имеющие смысл названия объектов и представлений объектов. На решении именно этой задачи должны сосредоточиться разработчики. Внимательно и осторожно выбирайте названия объектов и представлений. Обязательно проводите тестирование на удобство применения с участием потребителей.

Как и в реальном мире, совокупность объектов (возможно, различных типов), образует некоторую среду (окружение) пользователя. Исходя из этого, большинство заданий пользователя могут быть представлены (описаны) как определенная комбинация взаимосвязанных объектов. Так, например, обработка текстового документа может быть описана как композиция операций, выполняемых над его элементами (отдельными словами, параграфами и т.д.). Благодаря такому подходу любые, сколь угодно сложные конструкции могут быть реализованы на основе небольшого числа базовых соглашений.

При условии последовательной и согласованной реализации этих соглашений для всего пользовательского интерфейса эффективность работы пользователя существенно возрастает. Кроме того, указанный подход способствует модульной, компонентно-ориентированной разработке приложения, т.е. новое задание может быть выполнено путем адаптации или рекомбинации тех же объектов.

В реальном мире объекты сохраняют свое текущее состояние до тех пор, пока оно не будет изменено под влиянием каких-либо внешних воздействий. Например, если вы, уходя из дому, закрыли окно, оно, скорее всего, останется в таком же состоянии до вашего возвращения. Это же правило должно быть справедливо и для объектов интерфейса. За исключением тех случаев, когда требуется явное указание пользователя на сохранение данных, все объекты окружения должны быть сохранены автоматически. Кроме того, должна сохраняться и визуальная информация о состоянии окружения, такая, например, как позиция курсора, расположение и размер окна, с тем чтобы это состояние могло быть восстановлено при последующем сеансе работы пользователя.

При всех достоинствах объектного подхода к разработке интерфейса, его использование само по себе не гарантирует требуемого качества интерфейса. Для создания эффективного пользовательского интерфейса необходимо дополнить объектный подход тщательным проектированием всех компонентов интерфейса с ориентацией на потребности потенциального пользователя.

Первым шагом в объектно-ориентированном проектировании интерфейса должен быть анализ целей пользователей и особенностей выполняемых ими заданий. При проведении такого анализа следует определить основные компоненты или объекты, с которыми взаимодействует пользователь, а также характерные особенности объектов каждого типа. Необходимо также выявить перечень операций, выполняемых над объектами, их влияние на состояние и свойства объектов.

После завершения анализа можно переходить к описанию возможных способов взаимодействия пользователя с объектами различных типов. На этом шаге выбирается форма визуального представления объектов. При этом следует иметь в виду, что визуальный образ объекта в зависимости от ситуации может изменяться. Например, контейнер может быть представлен и в виде пиктограммы, и в виде окна, отображающего содержимое этого контейнера.

Завершает процесс проектирования компоновка и размещение на экране визуальных элементов интерфейса.

Перспективы эволюции интерфейсов – деятельностно-ориентированные интерфейсы

Объектные интерфейсы практически всегда гораздо лучше интерфейсов имплементационных (собственно говоря, объектные интерфейсы – это следующая, после имплементационных интерфейсов, ступень эволюции). Поспорить с этим невозможно, но открытым остается вопрос – что лучше объектных интерфейсов? Какова следующая эволюционная ступень?

Это, вероятно, деятельностно-ориентированные интерфейсы (ДоИ, обратите внимание, что это наш, относительно условный термин). В отличии от объектных интерфейсов, в которых пользователю предоставляются объекты и свобода манипулирования ими, в ДоИ «строительными блоками» являются задачи пользователя. В качестве примера можно привести диалоговое окно создания нового документа в MS Word: создать можно не просто обобщенный документ, одинаково плохо решающий все задачи, но документ для текущей задачи пользователя (например, письмо).

Перед тем, как перечислять преимущества этого вида интерфейсов, нужно определить, чем несовершенны интерфейсы объектные. Недостатки их таковы:

1. Такие интерфейсы требуют от пользователя существенных когнитивных нагрузок и способности к абстрактному мышлению: пользователю необходимо транслировать цель своих действий в конкретный алгоритм использования объектов. Это требует определенных усилий, так что при прочих равных от этой работы пользователей следует освободить.
2. Объектные интерфейсы зачастую требуют от пользователя серьезных усилий по отвлечению от собственной предметной области и привлечения внимания к вопросу «компьютерной поддержки» своей деятельности: репрезентируемые объекты, будучи универсальными, не полностью соответствуют предметной области конкретного пользователя. Например, разные интерфейсы почтового клиента нужны пользователю, который получает 5 писем в день и пользователю, который получает 70 писем, хотя в обоих случаях объекты (письма) одни и те же.
3. Во многих случаях «объектность» интерфейса подталкивает его разработчика к совершению некорректных действий: в поиске объектов деятельности разработчики чаще всего ограничиваются правами доступа. Смена парадигмы может форсировать разработчиков более тщательно подходить к проектированию

функциональности и интерфейса (хотя и новая парадигма, вероятно, тоже будет подталкивать разработчиков к совершению некорректных действий).

Деятельностно-ориентированные интерфейсы этих недостатков, как правило, лишены, благодаря чему они оказываются более эффективными, чем объектные интерфейсы.

Частным, вырожденным, примером деятельностно-ориентированного интерфейса являются мастера (Wizards). Вместо того, чтобы показывать пользователю единое диалоговое окно со сложной конфигурацией элементов управления, пользователю выдается сравнительно простая последовательность экранов. В идеальном случае, когда каждый последующий экран зависит от действий пользователя на предыдущем экране, мастер оказывается чрезвычайно эффективным. В неидеальном случае, когда его экраны не связаны между собой, он всё равно оказывается более эффективным (незначительное снижение скорости работы компенсируется низкими когнитивными нагрузками пользователя и другими факторами).

Но мастера это только начало пути: деятельностно-ориентированного в них не так уж и много. В будущем интерфейсы, идя по пути большей ориентированности на задачи и деятельность пользователей, достигнут ещё большей эффективности, при этом:

1. Разделение между интерфейсами, предназначенными для профессионалов и непрофессионалов, будет увеличиваться (как это уже произошло во всех других областях техники).
2. Программы будут становиться все более и более специализированными: не просто текстовый редактор, а текстовый редактор для сугубо определенной группы пользователей, решающих определенные задачи.
3. Адаптивность интерфейсов будет увеличиваться: они будут все больше и больше учитывать характерные особенности работы конкретных пользователей.
4. Количество технических подробностей (файлы, папки, порты и т.п.) в большинстве интерфейсов будет резко сокращено, поскольку эти объекты не связаны напрямую с деятельностью пользователей.