А.С. Орлов

ОСНОВНЫЕ ПОЛОЖЕНИЯ ТЕХНОЛОГИИ ПРОЕКТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКИХ ИНТЕРФЕЙСОВ

Программные приложения разрабатываются для обеспечения работы пользователя, т.е. для того чтобы он с помощью компьютерной программы быстрее и качественнее решал свои производственные задачи. С точки зрения эргономики, самое важное в программе — создать такой пользовательский интерфейс (ПИ), который сделает работу эффективной и производительной, а также обеспечит удовлетворенность пользователя от работы с программой.

Эффективность работы означает обеспечение точности, функциональной полноты и завершенности при выполнении производственных заданий на рабочем месте пользователя. Создание ПИ должно быть нацелено на показатели эффективности:

- точность работы - определяется тем, в какой степени произведенный пользователем продукт (результат работы) соответствует предъявленным к нему требованиям. Показатель точности включает процент ошибок, которые совершил пользователь: число ошибок набора, варианты ложных путей или ответвлений, число неправильных обращений к данным, запросов и пр.;
- функциональная полнота - отражает степень использования первичных и обработанных данных, списка необходимых процедур обработки или отчетов, число пропущенных технологических операций или этапов при выполнении поставленной пользователю задачи. Этот показатель может определяться через процент применения отдельных функций в приложении;
- завершенность работы - описывает степень исполнения производственной задачи средним пользователем за определенный срок или период, долю (или длину очереди) неудовлетворенных (необработанных) заявок, процент продукции, находящейся на промежуточной стадии готовности, а также число пользователей, которые выполнили задание в фиксированные сроки.

Последовательность действий и набор инструментальных средств пользователя в ПИ должны быть подчинены технологическому процессу выполнения производственного задания. Не надо бояться сложности системы, надо избегать такого интерфейса, который не соответствует алгоритму решения пользовательских задач.

Необходимо тщательно продумать и осознать сценарий взаимодействия программы с пользователем, приведя его к оптимальной (относительно рассмотренных показателей) системе выполнения задач, и реализовать ПИ в соответствии с этой системой.

Для того чтобы разобраться в технологии решения задач пользователя, разработчику необходимо выяснить следующие моменты (исследуя деятельность пользователя):

1. Какая информация необходима пользователю для решения задачи?
2. Какую информацию пользователь может игнорировать (не учитывать)?
3. Совместно с пользователем разделить всю информацию на сигнальную, отображаемую, редактируемую, поисковую и результирующую.
4. Какие решения пользователю необходимо принимать в процессе работы с программой?

5. Может ли пользователь совершать несколько различных действий (решать несколько задач) одновременно?

6. Какие типовые операции использует пользователь при решении задачи?

7. Что произойдет, если пользователь будет действовать не по предписанному Вами алгоритму, пропуская те или иные шаги или обходя их?

Производительность работы отражает объем затраченных ресурсов при выполнении задачи как вычислительных, так и психофизиологических.

Дизайн ПИ должен обеспечивать минимизацию усилий пользователя при выполнении работы и приводить:

- к сокращению длительности операций чтения, редактирования и поиска информации;
- уменьшению времени навигации и выбора команды;
- повышению общей продуктивности пользователя, заключающейся в объеме обработанных данных за определенный период времени;
- увеличению длительности устойчивой работы пользователя и др.

Сокращение непроизводственных затрат и усилий пользователя - важная составляющая качества программного обеспечения.

Для оценки продуктивности используются соответствующие показатели, проверяемые специалистами по эргономике в процессе тестирования рабочего прототипа. Формирование таких показателей происходит в процессе определения требований к ПИ при изучении следующих вопросов:

1. Что от пользователя требуется в первую очередь?
2. Сколько информации, требующей обработки, поступает пользователю за период времени?
3. Каковы требования к точности и скорости ввода информации?
4. На какие операции пользователь тратит больше всего времени?
5. Чем мы можем облегчить работу пользователя при решении типовых задач?

Удовлетворенность пользователя от работы тесно связана с комфортностью его взаимодействия с приложением и способствует сохранению профессиональных кадров на предприятии Заказчика за счет привлекательности работы на данном рабочем месте.

Требования к удобству и комфортности интерфейса возрастают с увеличением сложности работ и ответственности пользователя за конечный результат. Высокая удовлетворенность от работы достигается в случае:

- прозрачной для пользователя навигации и целевой ориентации в программе. Главное, чтобы было понятно, куда идем, и какую операцию программа после этого шага произведет;
- ясности и четкости понимания пользователем текстов и значения окон. В программе должны быть те слова и графические образы, которые пользователь знает или обязан знать по характеру его работы или занимаемой должности;
- быстроты обучения при работе с программой, для чего необходимо использовать преимущественно стандартные элементы взаимодействия, их традиционное или общепринятое их расположение;
- наличия вспомогательных средств поддержки пользователя (поисковых, справочных, нормативных), в том числе и для принятия решения в неопределенной ситуации (ввод по умолчанию, обход «зависания» процессов и др.).

Для оценки необходимого уровня удобства интерфейса также используются специальные опросники, формуляры, чек-листы, однако к данной работе лучше привлекать специалистов по эргономике.

Удобный интерфейс помогает пользователю справиться с усталостью и напряжением при работе в условиях высокой ответственности за результат.

Собственно процесс проектирования можно разделить на ряд этапов:

- определение необходимой функциональности системы;
- создание пользовательских сценариев;
- проектирование общей структуры;
- конструирование отдельных блоков;
- создание глоссария;
- сбор и начальная проверка полной схемы системы.

Этап определения необходимой функциональности системы нужен для того, чтобы, во-первых, не перегружать систему излишними функциями, а во-вторых, для корректного смыслового определения оставшихся функций. Обычно для решения этих задач используют копирование изделий конкурентов и учет пожеланий пользователей. Эти способы не лишены ряда существенных недостатков: во-первых, изделия конкурентов зачастую содержат множество ненужных или неудобных функций, во-вторых, пользователи не всегда могут грамотно выразить свои требования, и появляются существенные расхождения между желаемым и действительным. На мой взгляд, оптимальным является использование методов анализа целей и анализа действий пользователей.

Идеей, лежащей в основе данного метода анализа целей, является простое соображение, гласящее, что людям не нужны инструменты сами по себе, нужны лишь результаты их работы. Никому не нужен именно текстовый процессор, – нужна возможность с удобством писать тексты. Никому не нужна сама по себе программа обработки изображений – нужны уже обработанные изображения.

Это значит, что сами по себе функции не важны. Пользователям нужно средство вообще, делающее возможным выполнять определенную работу.

После того, как истинные цели пользователей установлены (и доказано, что таких пользователей достаточно много, чтобы оправдать создание системы), приходит время выбирать конкретный способ реализации функции, для чего используется второй метод анализа действий пользователей.

Один из вариантов этого метода - наблюдение за людьми, выполняющими свою задачу, пользуясь уже имеющимися инструментами, а именно системами конкурентов (если они есть) и предметами реального мира (поскольку очень немного новых действий появилось только после появления компьютеров).

Другим источником материала для анализа часто служит даже не наблюдение за людьми, но анализ результатов их работы – если оказывается, что результат работы практически не зависит от используемого инструмента, это значит, что нужна только та функциональность, которая оказала воздействие на результат (т.е. функции, которыми никто не воспользовался, не нужны).

Цель следующего этапа (Создание пользовательских сценариев) – написать словесное описание взаимодействие пользователя с системой, не конкретизируя, как именно проходит взаимодействия, но уделяя возможно большее внимание всем целям пользователей. Количество сценариев может быть произвольным, главное, что они должны включать все типы задач, стоящих перед системой, и быть сколько-нибудь реалистичными. Сценарии очень удобно различать по именам участвующих в них вымышленных персонажей.

Предположим, что необходимо разработать сценарии для будущей почтовой программы. Судя по всему, для этой задачи необходимо три сценария:

1. Боб запускает почтовую программу. Он включает процесс скачивания новой почты. Получив почту, он читает все сообщения, затем часть их удаляет, а на одно сообщение отвечает. После чего выключает почтовую программу.

2. Джек делает активным окно уже открытой почтовой программы и включает процесс скачивания новой почты. Получив почту, он ее читает. Одно сообщение он пересылает другому адресату, после чего удаляет его, а еще одно печатает. После чего переключается на другую задачу.

3. Пришло новое сообщение, и Алиса восприняла соответствующий индикатор. Она делает активным окно почтовой программы и открывает полученное сообщение. Она читает его, после чего перемещает его в другую папку. После чего переключается на другую задачу.

Польза этих сценариев двояка. Во-первых, они будут полезны для последующего тестирования. Во-вторых, сам факт их написания обычно (если не всегда) приводит к лучшему пониманию устройства проектируемой системы, побуждая сразу же оптимизировать будущее взаимодействие.

Дело в том, что на таких сценариях очень хорошо заметны ненужные шаги, например, в третьем сценарии гипотетическая Алиса после получения индикатора не смогла сразу же открыть новое сообщение, но должна была открыть окно системы, найти нужное сообщение, открыть его и только тогда прочесть. Понятно, что от этих ненужных этапов смело можно избавиться уже на этой, весьма ранней, стадии проектирования.

Следующий этап - проектирование общей структуры - состоит из двух параллельно происходящих процессов: выделения независимых функциональных блоков системы и определения связи между ними. Под отдельным функциональным блоком понимается функция/группа функций, связанных по назначению или области применения.

На данном этапе очень многое зависит от особенностей проектируемой системы. Тем не менее, можно рекомендовать избегать помещения в один блок более трех функций, поскольку каждый блок в результирующей системе будет заключен в отдельный экран или группу управляющих элементов. Перегружать же интерфейс опасно.

Результатом этой работы должен быть список блоков с необходимыми пояснениями.

Этап конструирования отдельных блоков является самым сложным и трудоемким, однако он подробно описан в специализированной литературе [1]. Его результатом является полный набор эскизов для всех диалоговых и документных окон разрабатываемого продукта.

На этапе "Создание глоссария" требуется зафиксировать все используемые в системе понятия (выписать текст с кнопок, названий режимов и т.п.). Потенциальные пользователи должны ознакомиться с этим списком и сообщить, как они понимают те или иные термины или команды. В случае несоответствий следует откорректировать глоссарий (как правило, неквалифицированные пользователи подбирают более простые и понятные термины).

На заключительном этапе сбора и начальной проверки системы производится объединение общей схемы системы и планов отдельных экранов с учетом

замечаний, выявленных на этапе "Создание глоссария". Полученная схема позволяет отследить изменение интерфейса системы в процессе вызова пользователем различных ее функций, а также определить, какие режимы недоступны на каждом заданном режиме и т.п., т.е. найти и зафиксировать возможные логические ошибки. Проанализированная и отредактированная схема теперь может использоваться для создания прототипа приложения его дальнейшей разработки.

ЛИТЕРАТУРА

1. Card, Moran, and Newell, «The Psychology of Human Computer Interaction», Erlbaum, 1983.
2. *Головач В. В.* Дизайн пользовательского интерфейса. http://www.uibook1.ru
3. *Андреев Д.* Что надо знать разработчику об эргономике программного продукта. http://www.usability.ru/Articles/instruction.htm
4. *Астапов М.* Пользовательский интерфейс программ. http: //delphi. vitpc. com/article/interface.htm

В.В. Булаев, И.Н. Котов, Б.А. Телеснин

АНАЛИЗ СОВРЕМЕННЫХ СРЕДСТВ РАЗРАБОТКИ СУБД-ПРИЛОЖЕНИЙ

Современные средства разработки информационных систем ориентированы на широкую поддержку различных СУБД, как настольных, так и серверных. Построение эффективных и надежных с точки зрения сохранности и защиты данных многопользовательских информационных систем, как правило, производится с использованием последних. Однако спектр средств разработки достаточно широк, хотя и обладает достаточно сходными возможностями в плане создания СУБД-приложений. Целью данного исследования является анализ наиболее популярных сред разработки с точки зрения легкости создания, изменения и производительности конечного приложения. В качестве предмета анализа выступают: Borland Delphi 5, Microsoft Visual Basic 6, Java (Sun JDK 1.3) для использования СУБД MS SQL Server 2000.

Для начала рассмотрим, какие средства взаимодействия с БД предлагаются данными средствами. С выходом Delphi 5 кроме устоявшейся технологии BDE появилась и новая – ADOExpress. Технология ActiveX Data Objects (ADO) – это разработка Microsoft. Она предоставляет упрощенный способ для доступа к данным, основанный на OLE DB, так как программирование непосредственно на OLE DB уровне достаточно сложно. Некоторые из наиболее интересных возможностей ADO связаны с использованием курсоров со стороны клиента. Можно, например, скопировать полностью набор данных на клиентский компьютер и выполнить ряд операций на этом кэше, включая сортировку, фильтрацию и редактирование данных. Можно даже скопировать часть данных в локальный файл и работать с ним в режиме offline. ADO и BDE – не единственная альтернатива для доступа к данным в Delphi. Есть и другие компоненты наборов данных, производимых Borland и третьими фирмами для непосредственного доступа к SQL серверам, таким, в частности, как и MS SQL Server. Есть также альтернативы для ADO - это разработки, которые предоставляют прямой доступ к DAO.

Microsoft Visual Basic 6.0 обеспечивает простое создание приложений, ориентированных на данные. Visual Basic 6.0 поддерживает универсальный интер-