**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ

1. Анализ предметной области

1.1 Оценка и изучение предметной области

1.2 Определение необходимой функциональности системы

1.2.1 Анализ целей пользователей

1.2.2 Анализ действий пользователей

1.3 Создание пользовательских сценариев

2. Высокоуровневое проектирование

2.1 Проектирование структуры экранов системы

2.2 Проектирование навигационной системы

2.3 Проектирование структуры справочной системы

3. Низкоуровневое проектирование

3.1 Проектирование экранов клиентской части

3.2 Проектирование экранов администраторской части

3.3 Тестирование

3.4 Экспертная оценка

4. Построение прототипа пользовательского интерфейса

4.1 Этапы построения прототипа

4.1.1 Бумажный

4.1.2 Презентационный

4.1.3 Псевдореальный

4.1.4 Реальный

5. Тестирование и модификация прототипа

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

**ВВЕДЕНИЕ**

Одной из важнейших задач, практически всегда стоявших перед человечеством, является сохранение информации во времени и пространстве. Со времён возникновения книгопечати основной формой хранения и распространения информации являются книги (печатные издания), а главными средствами хранения и доступа к информации стали библиотеки.

Сохранение и использование рукописных и печатных документов достаточно хорошо освоено. Здесь имеется богатый опыт, результаты исследовательской и практической работы многих поколений специалистов. Но очевидно, что с постоянным увеличением объёмов информации, работа с ней (такая как хранение, поиск, учёт и т.д.) становится всё сложнее. Развитие вычислительной техники позволило сохранять и распространять информацию в электронной форме. Это играет революционную роль в истории человечества, сравнимую с изобретением книгопечати.

Электронная форма позволяет хранить информацию в более надёжном, компактном и удобном виде, увеличивается скорость и простота её распространения. Немаловажно и то, что в электронной форме, в отличие от печатной, информацией можно свободно манипулировать. В связи с этим, количество электронных публикаций с каждым годом растёт.

Виртуальная библиотека - это информационная система, позволяющая надёжно сохранять и эффективно использовать разнообразные коллекции электронных документов, локализованных в самой системе, а также доступных ей через телекоммуникационные сети.

Существенное развитие работы по электронным библиотекам получили на рубеже 90-х годов, когда появились адекватные средства вычислительной техники и информационные технологии, обеспечивающие надёжное сохранение, оперативную обработку и эффективное использование больших массивов разнородной информации, прежде всего текстовой. Именно в это время в ряде стран стали подготавливаться проекты цифровых библиотек.

Основные задачи виртуальной библиотеки:

* интеграция разнородных информационных ресурсов;
* регистрация (публикация) новых данных;
* хранение и предоставление доступа к таким данным;
* координация других электронных коллекций по профилю данной библиотеки.

Под интеграцией информационных ресурсов понимается их объединение с целью использования (с помощью удобных и унифицированных пользовательских интерфейсов) различной информации с сохранением её свойств, особенностей представления и пользовательских возможностей манипулирования с ней. При этом объединение ресурсов не обязательно должно осуществляться физически, оно может быть виртуальным. Главное, чтобы оно обеспечивало пользователя возможностью воспринимать доступную информацию как единое целое. Новые технологии, разрабатываемые в этой области, прежде всего и ориентированы на решение этой важнейшей задачи. В частности, цифровые библиотеки принесли с собой новый, более высокий уровень семантического описания.

Эффективная навигация в электронной библиотеке понимается как возможность пользователя находить интересующую его информацию с наибольшей полнотой и точностью при наименьших затратах усилий во всём доступном информационном пространстве. При таком подходе хорошо известные информационные поиски, используемые в информационных системах и базах данных, являются частными случаями навигационных средств.

На сегодня, печатная информация является основным источником формирования цифровых библиотек.

**1. Анализ предметной области**

Предметная область - это материальная система или система, характеризующая элементы материального мира, информация о которой хранится и обрабатывается. Предметная область рассматривается как некоторая совокупность реальных объектов и связей между ними. Каждый объект обладает определённым набором свойств (атрибутов).

**1.1 Оценка и изучение предметной области**

Для того чтобы адекватно оценить ресурсы (время, деньги, количество экспертов), которые будет необходимо потратить на разработку (переработку) интерфейса, необходимо четко представлять себе объем информации, с которой следует ознакомиться.

Чтобы предлагать адекватные интерфейсные решения заказчику, необходимо иметь четкое представление о предметной области системы. Предметная область изучается по литературе. Наряду с этим, как правило, проводится серия интервью с экспертами для выяснения основных аспектов и характеристик предметной области.

После получения необходимых знаний об интересуемой предметной области, следует экспертная оценка текущего интерфейса системы. Вместе с собственными «жалобами» заказчика на текущую версию системы, результаты этого этапа составляют основное содержание работы над проектом (экспертная оценка часто обнаруживает проблемы, которые заказчику не видны, маскируясь под другие). Проблемы, выявленные на данном этапе, должны быть решены в новом интерфейсе, а удачные решения желательно сохранить, чтобы имеющимся пользователям не пришлось полностью переучиваться (и чтобы сократить затраты на переделку). Основное внимание уделяется не удачным решениям в существующем интерфейсе. Если на этом этапе проводилось юзабилити-тестирование текущей версии, отчет содержит краткие протоколы и перечень выводов исследования.

**1.2 Определение необходимой функциональности системы**

На первом этапе необходимо определить функциональность будущей системы. Это исключительно важный этап, поскольку именно функциональность будет определять весь интерфейс. Очень важно сознавать, что практически невозможно убрать из уже продающейся системы какие-либо функции. Во-первых, программы до сих пор продаются по функциям, т.е. чем больше список функций на коробке с программой, тем легче её продать, даже если большинство функций либо не работает, либо не нужно пользователям. Происходит это потому, что существенную часть тиража программы покупают новые пользователи, которые ничего не знают про истинное положение вещей. Во-вторых, имеющиеся пользователи обычно с исключительной неохотой переучиваются для использования новых функций взамен прежних (при этом еще и обижаются из-за того, что их инвестиции в обучение оказались выброшенными на ветер). Это значит, что ненужная функция системы будет кочевать из версии в версию, раздувая размеры программы, снижая надежность и быстродействие, и, что для нас более важно, портя интерфейс (при этом и длительность разработки возрастает). Несколько лучшая ситуация с сайтами - никого пока не удивляет, когда сайт после изменения дизайна почти не напоминает прежний. Так что оптимальным вариантом работы почти всегда является проектирование функциональности сразу на несколько версий вперед.

Традиционно требования к функциональности исходят от отдела продаж или от его аналога. Такие требования имеют два источника, а именно жалобы имеющихся клиентов и системы конкурентов. К сожалению, оба эти источника сомнительны.

Всегда может оказаться, что желание клиента получить новую функцию, обусловлено не реальной потребностью в ней, а собственно концептуальными проблемами системы. Системы же конкурентов страдают как от такого же непонимания проблемы, так и от множества других причин. Это значит, что прислушиваться к сторонним требованиям к системе, безусловно, следует, но рассматривать эти требования нужно не как директивы, но как признаки общего неблагополучия. Некой фирмой было разработано техническое задание к системе каталогизации изображений, рассчитанной на домашних пользователей, которая состояла из сведенного в таблицу списка всех функциональных возможностей всех конкурентов. В результате система получила огромное количество никому не нужных возможностей, что не только никак не помогло ей на рынке, но и безмерно удлинило срок её разработки.

Существует два принципиально разных подхода к определению функциональности системы. При первом подходе система снабжается максимальным количеством функций, при этом результаты многих из них являются суммой результатов других функций. При другом подходе все составные функции (метафункции) из системы изымаются. Ярким представителем первого подхода является CorelPhotoPaint, не менее ярким представителем другого - AdobePhotoShop.

Оба подхода имеют как недостатки, так и достоинства. Подход, при котором количество функций ограничено, позволяет упрощать интерфейс, но при этом требует от пользователя понимать, как из многих низкоуровневых функций «собирать»более сложные. Подход, при котором помимо низкоуровневых функций есть высокоуровневые, позволяет потенциально обеспечивать большую скорость работы (за счет отсутствия пауз между низкоуровневыми функциями), но зато требует от пользователя знаний о том, где эти высокоуровневые функции найти и как с ними работать, при этом они перегружают интерфейс.

Судя по всему, людям больше нравится пользоваться низкоуровневыми функциями, поскольку это позволяет добиваться более тонких и предсказуемых результатов. С другой стороны, доказательств этого не так уж много (сам факт того, что PhotoShop продается лучше, чем PhotoPaint, говорит не так уж о многом). Таким образом, выбор подхода не слишком прост. С другой стороны, остается возможность компромисса: всегда можно включить в систему средства автоматизации, чтобы пользователи получали возможность создавать (и распространять) свои метафункции, каковой подход как раз и применен в PhotoShop.

Так как всё-таки определить нужную функциональность? Современная наука выдвинула два основных способа, а именно анализ целей и анализ действий пользователей. Эти способы фактически не конфликтуют друг с другом, более того, в процессе определения функциональности желательно использовать оба.

**1.2.1 Анализ целей пользователей**

Идеей, лежащей в основе данного метода, является простое соображение, гласящее, что людям не нужны инструменты сами по себе, нужны лишь результаты их работы. Никому не нужен молоток, если не нужно забивать гвозди. Никому не нужен текстовый процессор, но нужна возможность, с удобством, писать тексты. Никому не нужна программа обработки изображений - нужны уже обработанные изображения.

Разработчику необходимо четко осознавать, что пользователям не нужны инструменты сами по себе, нужны лишь результаты их работы. Никому не нужен текстовый процессор - нужна возможность с удобством писать тексты. Никому не нужна программа обработки изображений - нужны уже обработанные изображения. Это значит, что сами по себе функции никому не нужны и не важны. Людям нужно средство, позволяющее выполнять какую-либо работу.

Ни в коем случае нельзя дать обмануть себя ненужной конкретикой, т.е. описанием того, какова должна быть будущая функциональность. Как правило, одного и того же результата можно добиться несколькими разными способами, при этом важно не только реализовать какой-либо способ, но и выбрать лучший.Результатом этого процесса должен являться список целей.

Анализ целей пользователя виртуальной библиотеки: «Виртуальная библиотека должна иметь список доступных книг, из которых человек сможет выбирать интересующие его. В ней обязательно должна присутствовать поисковая система (для более удобной навигации). Помимо клиентской части приложения должен быть и некий редактор, позволяющий манипулировать информацией цифровой библиотеки».

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.

**1.2.2 Анализ действий пользователей**

Достижение почти всех целей требует от пользователей совершения определенных действий. Разумеется, эти действия могут различаться при разных способах достижения. В сколько-нибудь сложных интерактивных системах сами по себе выбранные стратегии действий влияют на требования к функциональности.

В компьютерных системах взаимодействие сложнее, чем в реальности, при этом логический анализ неприемлем.Единственным выходом является банальное 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). Неплохим 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).

Тут очень важно избежать эгоцентризма. Очень трудно отказаться от мысли «если это нужно мне, это нужно и многим». Возможно, что и нужно. А возможно, что и нет. Единственным же способом проверить, нужна функция или нет, является наблюдение за пользователями и анализ их действий.

Как уже было сказано, обычно есть несколько разных способов реализации одной и той же функции. Анализ действий пользователей как раз и позволяет определить, какой именно способ следует реализовывать. Поскольку на этом этапе мы узнаём, какая именно функциональность нужна для каждого варианта, можно избрать верный путь по правилу «чем меньше действий требуется от пользователя, тем лучше» (благо компьютер есть, прежде всего, великое средство автоматизации). Не стоит забывать и про другое правило: чем меньше функций, тем легче их сделать.

**1.3 Создание пользовательских сценариев**

Цель этого этапа - написать словесное описание взаимодействия пользователя с системой, не конкретизируя, как именно проходит взаимодействие, 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 будут полезны для последующего тестирования, поскольку тестируется не выполнение абстрактных задач, а выполнение конкретных, входящих в эти сценарии, операции. Да и сам факт их написания обычно, хоть и не всегда, приводит к лучшему пониманию устройства проектируемой системы, побуждая сразу же оптимизировать будущее взаимодействие.

Сценарий взаимодействия пользователя с виртуальной библиотекой: «Пользователь открывает программу, выбирает книгу из каталога и начинает её чтение».

Сценарий взаимодействия администратора с программой: «Администратор открывает редактор, выбирает существующую книгу для редактирования или добавляет новую в список».

**2. Высокоуровневое проектирование**

**2.1 Проектирование структуры экранов системы**

На этом этапе, основываясь на сценариях работы и ролях пользователей, формируется структура экранов системы, т.е. определяется количество экранов, функциональность каждого из них, навигационные связи между ними, формируется структура меню и других навигационных элементов.

По сути, на этом этапе выделяются отдельные функциональные блоки. Под функциональными блоками будет подразумевать функцию или группу функций, связанных по назначению или области применения в случае программы и группу функций/фрагментов информационного наполнения в случае сайта.

Программное обеспечение «Виртуальная библиотека» будет состоять из двух частей: клиентской (Таблица 2.1) и администраторской (Таблица 2.2).

Таблица 2.1 - функциональные блоки клиентской части приложения.

|  |
| --- |
| Основной экран  Навигация между различными наименованиями книг |
| Поиск необходимой книги |
| Сортировка доступных книг по категориям |
| Экран краткой информации о книге  Описание книги |
| Получение полного содержания книги |
| Экран содержания книги  Текст книги |

Таблица 2.2 - функциональные блоки администраторской части приложения.

|  |
| --- |
| Основной экран  Навигация между различными наименованиями книг |
| Поиск необходимой книги |
| Сортировка доступных книг по категориям |
| Добавление книги в список |
| Редактирование книги из списка |
| Удаление книги из списка |
| Экран добавления и редактирования книги  Заполнение и изменение информации о книге |

Существует три основных вида связи между блоками. Это «логическая связь», «связь по представлению пользователей» и «процессуальная связь».

Логическая связь. Логическая связь определяет взаимодействие между фрагментами системы с точки зрения разработчика. Полученные связи очень существенно влияют на навигацию в пределах системы (особенно, когда система многооконная). Поэтому чтобы не перегружать интерфейс, стоит избегать как слишком уж отдельных блоков (их трудно найти), так и блоков, связанных с большим количеством других.

На этом этапе выделяются объективные критерии оценки эргономичности интерфейса, такие как показатели эффективности, продуктивности, удовлетворенности пользователей (на более ранних этапах выделить эти критерии невозможно).

Связь по представлению пользователей. Пользователи имеют свое мнение о системе, и это мнение тоже является важным видом связи. В информационных системах, когда необходимо гарантировать, что пользователь найдет всю нужную ему информацию, необходимо устанавливать связи между блоками, основываясь не только на точке зрения разработчика, но и на представлениях пользователей. Дело в том, что самый распространенный способ поиска, а именно поиск по классификации признаков, работает только в том случае, когда пользователи согласны с принципами этой классификации. Большинство же понятий однозначно классифицировать нельзя из-за наличия слишком большого количества значимых признаков. Также проблема состоит в том, что реальный классификационный признак может отличаться от широко распространенного.

Из этой ситуации есть выход. Существует простой способ классификации - способ карточной сортировки. Все понятия, которые требуется классифицировать, пишутся на карточках из расчета «одно понятие - одна карточка». После чего группе пользователей из целевой аудитории предлагается эти карточки рассортировать (при этом каждый субъект получает свой набор карточек). Получившиеся кучки из карточек нужно разобрать на составляющие и свести результаты от разных субъектов в один способ классификации.

Процессуальная связь. Процессуальная связь описывает не вполне логичное, но естественное для имеющегося процесса взаимодействие. Установление качественной процессуальной связи обычно довольно трудная задача, поскольку единственным источником информации является наблюдение за пользователем. В то же время установление такой связи дело исключительно полезное. Зачем, например, рисовать на экране сложную систему навигации, если точно известно, к какому блоку пользователь перейдет дальше? В этом смысле зачастую оправдано навязывать пользователю какую-либо процессуальную связь, жертвуя удобством, зато выигрывая с скорости обучения (поскольку пользователю приходится думать меньше). Жестко заданная связь позволяет также уменьшить количество ошибок. Классическим примером жестко заданной процессуальной связи является устройство мастеров, при котором пользователя заставляют нажимать кнопку «Далее».

**2.2 Проектирование навигационной системы**

На основе разработанной ранее структуры экранов на этом этапе выбирается наиболее адекватная навигационная система и разрабатывается её детальный интерфейс.

Навигационная система показывает механизм распределения функций и задач между окнами программы. Навигационная система определяет, каким образом пользователи смогут перемещаться как между различными задачами, так и внутри отдельной задачи. Например, можно ли будет оставить частично завершенную задачу и начать другую.

Как правило, на этом этапе не создается отдельного отчета; разработанный интерфейс в дальнейшем описывается в отчете, посвященному низкоуровневому проектированию.

Навигация в программе будет осуществляться посредством мышки, путём нажатия на графические элементы - кнопки (для перехода между экранами).

**2.3 Проектирование структуры справочной системы**

Разрабатывается справочная система (к настоящему этапу уже известна вся информация, необходимая для выработки структуры справочной системы, не просто описывающей интерфейс, но и помогающей пользователю решать его задачи).

Множество систем, ни при каких обстоятельствах не могут быть используемы без подготовки, независимо от качества их интерфейса. Система автоматизации, например, может быть эффективно использована только в том случае, когда пользователь этой системы понимает суть автоматизируемых процессов. Для этого пользователя необходимо снабдить пониманием устройства системы. Это значит, что концепции и суть сложной системы могут быть безболезненно вынесены из интерфейса в документацию, освобождая ресурсы дизайнера. С другой стороны, зачастую описание концепций влияет на их реализацию в системе, особенно в ситуациях, когда качество обучения работе с системой является важнейшей составляющей общего качества. Это наблюдение вполне подтверждается опытом. Например, Джеф Раскин, Отец Макинтоша, изначально был начальником отдела документации. После того, как он обнаружил, что имеющуюся систему невозможно приемлемо описать, он создал новую, описывающуюся хорошо. Побочным свойством новой системы, компьютера Макинтош, было то, что его интерфейс был понятен и удобен в работе.

Документация является частью интерфейса, причем в сложных системах это большая часть.

К сожалению, создание справочной системы воспринимается в нашей стране как сравнительно необязательный процесс, нужный исключительно для утяжеления коробки с готовым продуктом. От этого документация почти всегда пишется после того, как система создана.

**3. Низкоуровневое проектирование**

На данном этапе разрабатываются интерфейсы конкретных экранов системы (состав, взаимное расположение и поддерживающие текст интерфейсных элементов).

**3.1 Проектирование экранов клиентской части**

Основной экран. Как уже было отмечено ранее, основной экран в приложении должен отображать список доступных книг, иметь функцию сортировки информации, а также предоставлять пользователю функцию поиска (Рисунок 3.1).

Чтобы упорядочить наименования книг по автору, достаточно нажать на заголовок столбца «Автор» (Рисунок 3.2). При этом вокруг элемента появится рамка. Всего доступно 3 группы сортировки: по автору, по названию книги и по жанру.

Наибольшее пространство в главной форме принадлежит списку книг (Рисунок 3.3), справа от которого расположена вертикальная полоса прокрутки, дающая возможность перемещаться по этому списку. Активный элемент (при наведении мышки) подсвечивается зелёным цветом.

Форма поиска (Рисунок 3.4) позволяет искать пользователю интересующие его книги по списку.

Экран информации о книге. Данный экран предоставляет краткую информацию о книге, такую как: название, автор, издательство и небольшое описание (Рисунок 3.5).

Внизу формы расположена пара кнопок. Кнопка «Читать» откроет содержание книги, а кнопка «Каталог» вернёт пользователя к списку литературы.

Экран содержания книги. На этомэкране отображается содержание книги (Рисунок 3.6). Навигация по нему осуществляется с помощью полосы вертикальной прокрутки.

Для более удобной навигации, на этой форме присутствует поле поиска. Чтобы вернуться обратно к списку литературы, следует нажать на кнопку «Каталог».

**3.2 Проектирование экранов администраторской части**

Основной экран. Основной экран администраторской части несильно отличается от экрана клиентской. Помимо уже упомянутых ранее элементов, на нём появилось пара новых (Рисунок 3.7).

Около поля поиска находится кнопка «Добавить», предназначенная для добавления новых книг в список. Для изменения или удаления уже существующих записей в таблице существует 2 кнопки, расположенные левее от наименования - «Удалить» (иконка в виде крестика) и «Изменить» (с иконкой в виде документа).

Экран изменения и редактирования. На этом экране осуществляется заполнение информации о книге, перед её добавлением в таблицу или при изменении уже существующей (Рисунок 3.8).

Кнопка«Сохранить» - заканчивает редактирование информации о книге. Кнопка «Назад» возвращает пользователя к каталогу. «Изменить картинку» позволяет указать изображение для книги, которое впоследствии будет отображаться в краткой информации о ней. Кнопка «Выбрать файл содержания» позволяет указать файл, в котором находится текст книги

**3.3 Тестирование**

На основе объективных критериев успеха интерфейса и сценариев действий пользователей разрабатываются тестовые задания, которые выполняются пользователями с фиксацией всех значимых характеристик их деятельности. После этого выполняется подсчет соответствующих показателей и сравнение их с заданными. На основании полученных данных интерфейс либо дорабатывается, либо считается разработанным.

В процессе проектирования полезно зафиксировать все используемые в системе понятия. Для этого нужно просмотреть все созданные экраны и выписать из них все уникальные понятия (например, текст с кнопок, названия элементов меню и окон, названия режимов и т.д.). После этого к получившемуся списку нужно добавить определения всех концепций системы (например, книга или изображение).

После этого необходимо этот список улучшить. Для этого необходимо:

* уменьшить длину всех получившихся элементов;
* показать этот список любому потенциальному пользователю системы и спросить его, как она понимает каждый элемент. Если текст какого-то элемента воспринимается неправильно, его нужно заменить;
* уменьшить длину всех получившихся элементов;
* проверить, что одно и то же понятие не называется в разных местах по-разному;
* проверить текст на совпадение стиля с официальным для выбранной платформы (если вы делаете программу, эталоном является текст из MS Windows);
* уменьшить длину всех получившихся элементов;
* убедится, что на всех командных кнопках стоят глаголы-инфинитивы (Отменить, Удалить, Отправить).

Последней задачей перед построением прототипа является проверка внутренней логики системы. Дело в том, что всегда существует вероятность того, что вы что-то забыли или спланировали неправильно. Как уже было сказано, исправить эти ошибки лучше всего до построения прототипа (даже первой его версии). Конечно, многие структурные ошибки нельзя найти никакими методами, кроме длительного логического анализа. С другой стороны, практика показывает, что почти все найденные ошибки будут существенными. Так что лишняя проверка не повредит.

Для финальной проверки схемы пригодятся разработанные ранее пользовательские сценарии. Не глядя на схему, необходимо подробно описать, как все вымышленные пользователи будут взаимодействовать с системой, не пропуская ни одного элемента управления. После чего сверить полученный текст со схемой.

Тут возможно три варианта развития событий: либо вы обнаружите, что вы что-то забыли задокументировать в схеме, либо обнаружите, что свеженаписанный рассказ значительно лучше схемы, вероятнее же всего, что и то и другое произойдет одновременно. На четвертый вариант, а именно на полное отсутствие проблем, рассчитывать, как правило, не стоит.

**3.4 Экспертная оценка**

Весьма эффективным средством оценки получающегося интерфейса является его экспертная оценка. Часто оказывается, что сравнительно дорогое тестирование показывает то, что было бы легко видно постороннему, тем более вооруженному опытом и квалификацией, взгляду. Хотя экспертная оценка не может быть полноценной заменой тестирования, она обладает одним существенным преимуществом - для её проведения не требуется прототип. Это значит, что эксперт может быть приглашен на ранних стадиях работы, когда польза от обнаружения ошибок максимальна.

Для проведения экспертной оценки нужно знать следующее:

* разные люди обнаруживают разные ошибки. Это значит, что метод работает лучше, когда количество экспертов больше единицы;
* лучше привлекать несколько экспертов не одновременно, но последовательно;
* чем больше информации о проектируемой системе будет предоставлено эксперту, тем более сложные проблемы он сможет выявить;
* нельзя требовать от эксперта работы по весу. В большинстве случаев результатом его работы будут одна или две страницы текста (поскольку описание одной проблемы требует обычно всего двух или трех предложений). Если от эксперта будет требоваться объемный результат работы, он включит в него много несущественных подробностей;

**4. Построение прототипа пользовательского интерфейса**

Создание максимально эффективного прототипа интерфейса является чрезвычайно важной задачей. Прототип должен хорошо выглядеть, чтобы понравиться заказчику и не вызвать вопросов у субъектов тестирования, он должен быть максимально дешев, максимально полон и, что немаловажно, должен с легкостью обновляться.

При создании прототипа наиболее частой ошибкой является чрезмерное наведение глянца и вообще стремление сделать прототип возможно более похожим на результирующую систему. В самом таком подходе нет ничего плохого (всё равно определенные части прототипа приходится делать максимально совершенными), проблема в том, что в большинстве случаев прототип после тестирования оказывается неправильным. Его приходится переделывать, причем иногда полностью, при этом все вложенные в прототип ресурсы (в основном время) оказываются выброшенными на ветер.

К счастью, требования к прототипу изменяются со временем. Сначала наиболее актуальными его свойствами являются скорость создания и простота модификации. Эти свойства позволяют быстро разработать и проверить несколько версий интерфейса, при этом ещё и исправить значительную часть ошибок. Только затем на первый план выходят функциональность и эстетичность, простота же модификации становится уже не так важна, поскольку с каждой новой исправленной ошибкой снижается вероятность того, что прототип придется полностью переделывать при обнаружении новой ошибки.

Поэтому всегда правильно делать прототип настолько похожим на результирующую систему, насколько версия прототипа поздняя. Первый прототип стоит делать максимально примитивным. Только после того, как тестирование подтверждает его правильность, стоит делать более детализированный прототип.

**4.1 Этапы построения прототипа**

Существуют четыре версии прототипа: бумажная, презентационная, псевдореальная и реальная. Особый интерес представляют первые две. Третья и четвертая используются редко, т.к. не сильно отличаются от готовой программы.

**4.1.1 Бумажный**

Необходимо нарисовать на бумаге все экраны и диалоговые окна (или распечатать соответствующие части схемы). Нужно только убедиться, что все интерфейсные элементы выглядят единообразно и сколько-нибудь похоже на реальные (Рисунок 4.1).

Эта распечатка и является первым прототипом. На нём вполне можно тестировать восприятие системы пользователем и её основную логику.

Достоинствами бумаги являются исключительная простота и скорость рисования (Рисунок 4.2). Польза начального прототипирования на бумаге заключается, во-первых, в простоте и скорости рисования, во-вторых, в исключительной простоте модификации по результатам тестирования, а в-третьих, в относительной простоте привлечения представителей целевой аудитории. Значительно легче завлечь субъекта к письменному столу, нежели к компьютеру, на котором надо что-то запускать, ждать, пока запустится и так далее. Кроме того, бумага помогает не думать о том, как интерфейс выглядит, позволяя сосредоточиться на том, как интерфейс работает.

При рисовании прототипа на бумажке очень важно не стараться нарисовать его так, чтобы он был красив, например, не нужно стараться рисовать линии прямыми. На понимание работы интерфейса это не повлияет, зато здорово замедлит работу. Красоту же всё равно придется выбрасывать, когда будет нарисована новая версию. Так что основным критерием живописности должна стать скорость работы.

Элементы интерфейса, которые нельзя нарисовать однозначно (например, раскрывающиеся списки, у которых значения до поры скрыты) эффективнее всего рисовать неоднозначными, важную же информацию из них лучше всего словами писать на полях.

Разумеется, значение слова «версия», весьма условно. В действительности после обнаружения каждой ошибки схема и прототип исправляются, а тестирование продолжается уже на новом прототипе. Так что на этом этапе прототип может пережить множество исправлений и, соответственно, много версий (Рисунок 4.3).

**4.1.2 Презентационный**

После исчерпания возможностей бумажной версии прототипа стоит создать новую версию. Для этого точно так же рисуется интерфейс, но уже не на бумаге, но в какой-либо презентационной программе (Рисунок 4.4). С этой версией прототипа можно тестировать значительно более сложное взаимодействие человека с системой, нежели с бумажной. С другой стороны, исправление найденных ошибок значительно более трудоемко.

Самым распространенным инструментом для создания прототипов второго типа является MS Visio. Некоторую конкуренцию ему могут составить MS PowerPoint и MacromediaFreeHand (и вообще любой иллюстративный пакет, обладающий возможностью работать с несколькими страницами).

При работе с Visio можно выбрать одну из двух возможностей: можно либо рисовать все экраны на одном листе, соединяя взаимосвязанные элементы управления и экраны линиями, либо рисовать каждый экран на отдельном листе, связывая экраны ссылками. Первый вариант хорош для вас, поскольку позволяет оценить интерфейс в целом, второй – для заказчика и субъектов тестирования, поскольку его легче понять. Как правило, превратить второй вариант в первый оказывается гораздо легче. Если рисовать в PowerPoint, каждой экран удобно создавать как отдельный слайд, а результат нажатия кнопок имитировать переходами между слайдами.

Среди многочисленных наборов заготовок в Visio есть и набор с элементами интерфейса Windows, однако эти заготовки выполнены довольно грубо, с огромным количеством лишних деталей. Пользоваться можно только заготовками для радиокнопок и чекбоксов (которые реализованы неплохо), а также заготовкой для полоски прокрутки. Поскольку создание собственных интерактивных заготовок непроблематично, рекомендуется создать свой набор и пользоваться им.

Большим достоинством Visio является возможность записывать результат в HTML-файл, что позволяет без проблем тестировать интерфейс на территории субъектов тестирования. Раньше это мог только PowerPoint, чем, во многом, и объяснялась его популярность в качестве инструмента для создания прототипов. Сейчас это умеет и Visio (сохранение в HTML начало нормально работать только в Visio 2001).

У многих разработчиков, особенно кто не понаслышке знаком с программированием есть соблазн воспользоваться для создания прототипа средствами быстрой разработки приложений (Delphi, VisualBasic и т.д.). В этом случае прототип интерфейса воспринимается лишь как вспомогательная оболочка над программным кодом. Как следствие возникают следующие недостатки:

* для тестирования прототипа на пользователях крайне желательно, чтобы он «немного работал», то есть, например, нажатие на кнопку вызывало другое окно или чтобы из выпадающего списка можно было выбрать значение. Так вот, в данном случае любое взаимодействие, как между отдельными элементами интерфейса, так и между различными формами реализуется только с помощью написания программного кода. Что совершенно незачем в данном контексте. Зачем усложнять то, что можно сделать проще? Ведь основной критерий создания прототипа это скорость;
* чрезвычайно сложно создать принципиально новый элемент интерфейса либо модифицировать уже имеющийся. Подобные задачи очень часто встречаются на практике, так как для «хорошего» интерфейса стандартных элементов, как правило, не хватает. Затраты на создание/модификацию собственного элемента управления в данных средах разработки нельзя назвать адекватными;
* каждый элемент управления здесь имеет несколько десятков различных свойств, тогда как для прототипизирования требуются лишь касающиеся внешнего вида настройки - шрифт, цвет, текст, размер;
* естественный недостаток: нельзя разрабатывать веб-интерфейс.

Фактически для большинства систем этой версии прототипа оказывается достаточно.

**4.1.3 Псевдореальный**

В тех случаях, когда в интерфейсе появляются нестандартные элементы или необходимо проверить реальную скорость взаимодействия пользователя с системой, создается еще одна версия прототипа – реально выглядящая, но лишенная каких-либо алгоритмов и, соответственно, не показывающая реальных данных. Делать этот вариант можно как в средах разработки, благо в них есть визуальные инструменты создания интерфейсов, так и в редакторах изображений, что обычно быстрее. Фактически при этом создаются фальшивые снимки экрана, на которых и производят тестирование(Рисунок 3.1). Понятно, что существенно модифицировать эти экраны затруднительно, так что лучше не увлекаться такой работой, не получив каких-либо гарантий ее правильности.

**4.1.4 Реальный**

Иногда необходимо тестировать взаимодействие пользователя не только с интерфейсом системы, но и с обрабатываемыми системой данными. Например, работая с графической программой, пользователь не только нажимает на экранные кнопки, но также создает и модифицирует изображения мышью. Область же редактирования данных зачастую вообще не содержит каких-либо визуальных интерфейсных элементов, из чего вовсе не следует, что интерфейса в ней нет, его, наоборот, много. Другой разговор, что счет в нем идет не на кнопки и переключатели, но на пиксели и миллисекунды.

Понятно, что создание прототипа в таких условиях не поможет, поскольку прототип вообще не будет отличаться от проектируемой системы. В таких условиях лучше всего написать нужные участки кода до написания всего остального, и проводить тестирование уже на реальной системе.

**5. Тестирование и модификация прототипа**

Какими бы не были совершенными логические соображения, приведшие к созданию интерфейса, всегда остается вероятность того, что интерфейс получился плохой, либо, что более вероятно, не такой хороший, каким бы он мог быть (Рисунок 5.1).

Необходимо иметь какие-либо подтверждения его работоспособности. К счастью, проверка качества интерфейса обычно непроблематична. Всё, что для этого нужно, это несколько пользователей средней квалификации, никогда не видевшие тестируемой системы, плюс прототип (разумеется, при наличии основательного бюджета можно развернуться и пошире, например, купить прибор, фиксирующий направление взгляда пользователя).

В литературе часто встречается мнение, что тестированием можно решить чуть ли не все проблемы интерфейса. Утверждение это сомнительно. Тестированием, скорее, можно определить слабые места интерфейса, но почти невозможно обнаружить сильные, поскольку они пользователями просто не замечаются, и совсем уж невозможно определить новые способы улучшения. Происходит это из-за того, что субъекты тестирования:

* не обладают всей необходимой информацией о системе;
* ничего не знают о проектировании интерфейсов;
* их мотивация существенно отличается от необходимой - вместо того, чтобы стремиться сделать хороший интерфейс, они стремятся оставить в этом интерфейсе свой след.

Вообще, слушать потребителей обычно неправильно. Разве мы спрашиваем канарейку, в какой клетке она хочет жить? Сюжет проамериканский автопром, например, стал уже частью истории бизнеса: все американские потребители в семидесятых годах дружно утверждали, что они хотят большие, мощные машины, при этом так же дружно покупая маленькие и маломощные японские автомобили. Или другой пример - в советское время измученные коммунизмом люди мечтали вовсе не об отпуске на Тенерифе, о котором они ничего не знали, но о финском хромированном смесителе, который поставил себе сосед - хотя Тенериф, безусловно, в качестве мечты интереснее.

В то же время, даже не слушая пользователей, обязательно нужно принимать во внимание их потребности, способности и предпочтения. Например, нередко дизайнер интерфейса знает о предметной области меньше, нежели будущие пользователи. В таких условиях потеря контакта с пользователями грозит крахом продукта, просто потому, что система оказывается неспособна решать задачи, о которых дизайнер ничего не знал. Чаще, однако, случается так, что уровень «компьютерной грамотности» дизайнера оказывается выше уровня аудитории. В этом случае все получается ещё хуже: дизайнер выбирает решения, которые обеспечивают эффективность работы, а потребителям нужны решения, которые они могут понять, в результате они оказываются неспособными воспользоваться системой (это совершенно нормальная ситуация, особенно в интернете). Например, замечено, что опытные пользователи (к которым относятся дизайнеры) создают значительно менее работоспособные иерархии меню, нежели пользователи начинающие.

Разумеется, если переоценка способностей реальных пользователей и незнание предметной области совпадают, результат бывает ещё хуже.

**ЗАКЛЮЧЕНИЕ**

Виртуальные библиотеки являются неотъемлемой частью настоящего и будущего. С каждым днём число электронных публикаций растёт, постепенно вытесняя бумажные издания.

Электронная форма позволяет хранить информацию в более надёжном, компактном и удобном виде. Значительно увеличивается скорость поиска нужной информации, простота её распространения.

При разработке программы, особое внимание стоит уделять её интерфейсу. Лаконичный, удобный, а главное понятный пользовательский интерфейс делает обучение работе с ним легким и быстрым, снижает время и затраты на обучение, техническую поддержку пользователей. Он способен повысить производительность труда пользователей, снизить количество человеческих ошибок, а значит, уменьшить количество времени на их исправление.