Тема 2

Технологии проектирования пользовательского интерфейса[1]

1. Эволюция процесса разработки пользовательского интерфейса
2. Проектирование взаимодействия
3. Этапы работы над пользовательским интерфейсом
4. Первоначальное проектирование: сбор функциональных требований
5. Проектирование общей структуры системы

«1. Эволюция процесса разработки пользовательского интерфейса.

Прежде чем мы будем говорить о проектировании, посмотрим, как развивался процесс разработки программного обеспечения в IT-индустрии согласно Алану Куперу (см. рис. 1).

В ранние дни развития IT-индустрии процесс разработки сводился к тому, что программисты вынашивали идею продукта, а затем создавали и самостоятельно тестировали его.

В более поздние времена к процессу стали подключаться профессиональные управленцы, их задачи сводились к оценке потребностей рынка и формулированию основных требований к разрабатываемому программному обеспечению.

С развитием IT-индустрии в самостоятельную дисциплину выделилось тестирование, а также широкое распространение получили графические интерфейсы пользователя, появилась необходимость разработки различных визуальных элементов, в связи с чем к процессу разработки ПО подключились графические дизайнеры и тестировщики.

В настоящее время используется подход к разработке программного обеспечения, при котором решения о возможностях продукта, его форме и поведении принимаются до начала дорогостоящей и сложной фазы создания продукта. Это обеспечивается включением в процесс разработки этапа проектирования, при этом эффективность проектирования во многом определяется выбранным стилем принятия решений. Можно выделить пять стилей принятия решений.

---

1. Источник: Брусенцова, Т. П. Проектирование интерфейсов пользователя: пособие для студентов специальности 1-47 01 02 «Дизайн электронных и веб-изданий» / Т. П. Брусенцова, Т. В. Кишкурно. — Минск БГТУ, 2019. — С. 68-94.

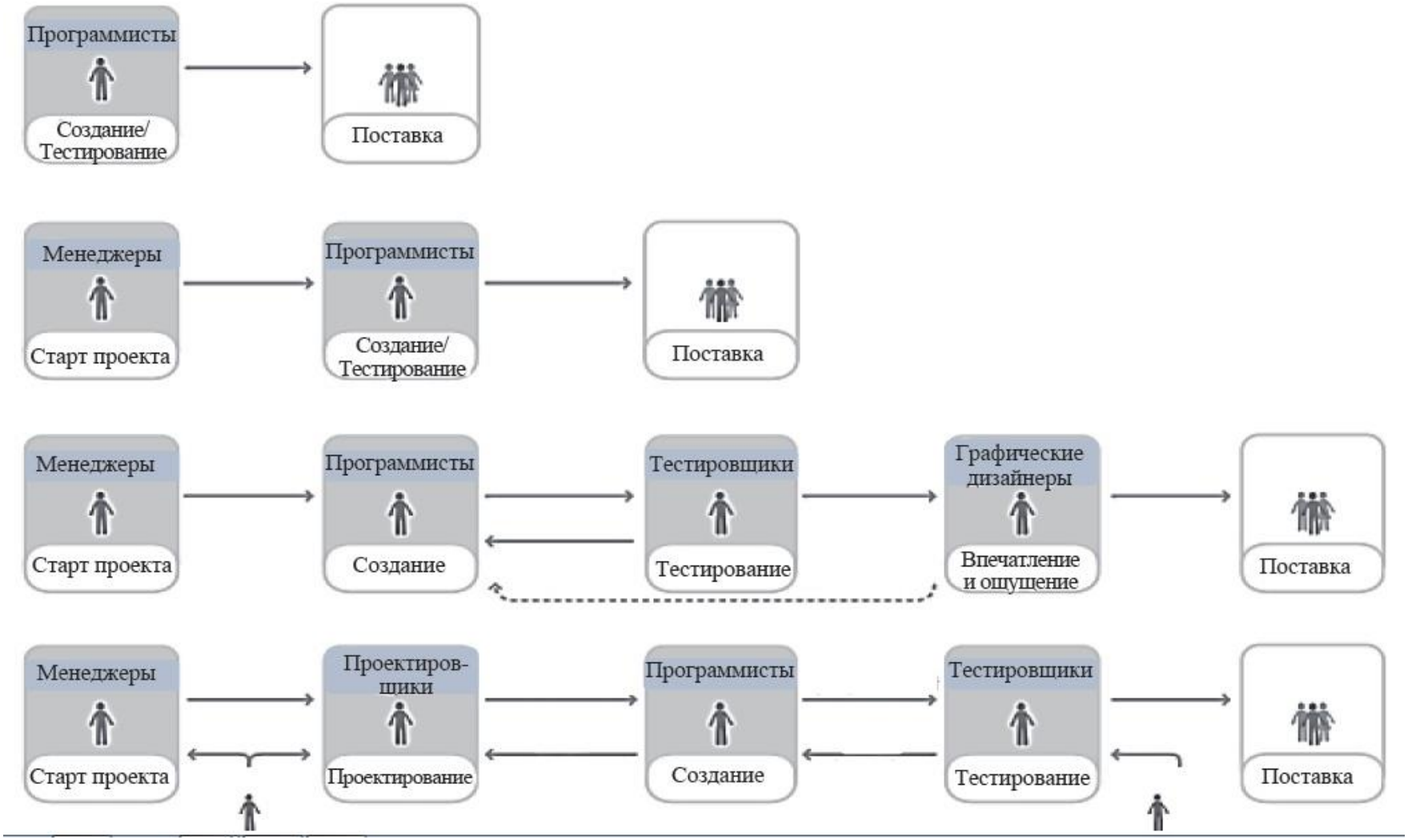

Рис. 1. Эволюция процесса разработки программного обеспечения

1. *Непреднамеренное» проектирование* команда сосредоточена на разработке и внедрении приложения, не задумывается об удобстве его использования.

2. *Проектирование «для себя»* основывается на опыте использования продукта членами команды, имеет большие шансы на успех, чем «непреднамеренное», хорошо, когда члены команды являются главными пользователями разрабатываемого продукта.

3. *Genius проектирование* основывается на опыте всех членов команды в проектировании подобных продуктов, хорошо работает, если уже есть опыт проведения предварительных исследований пользователей и сценариев их поведения с последующей проверкой соответствия дизайна ожиданиям пользователей.

4. *Проектирование, ориентированное на деятельность,* основывается на исследовании поведения пользователей. Для исследования часто применяются методики, основанные на деятельности, например, построение диаграмм последовательности операций (work flow diagrams) и ориентированные на задачи тестирования удобства использования.

5. *Проектирование, ориентированное на пользователя,* основывается на глубоком исследовании целей и нужд пользователей, контекста использования, дает возможность принимать детальные решения, которые были бы невозможны при применении других методов.

Стили перечислены в порядке возрастания объема исследований, которые необходимо провести команде проектировщиков для принятия решений. На первый взгляд, должно казаться, что оптимальным будет использование стиля, ориентированного на пользователей, но это не всегда так. Существуют проекты, для которых затраты времени и сил на проведение детального исследования пользователей не оправданы, с другой стороны, существуют проекты, которые без таких исследований просто провалились бы. Умение для каждого проекта выбрать наиболее эффективный стиль проектирования является одной из ключевых характеристик хорошего проектировщика. Наиболее эффективные команды должны владеть всеми пятью методами принятия решений в процессе проектирования и выбирать именно тот, который лучше бы соответствовал нуждам и целям проекта.

2. Проектирование взаимодействия

Проектирование интерфейса представляет собой довольно сложный и многоэтапный процесс, каждый этап которого состоит в свою очередь из отдельных ступеней. В общем случае весь процесс можно представить в виде четырех этапов (рис. 2.), направленных на

решение основной задачи — обеспечить оптимальное взаимодействие пользователя с системой.

Рис. 2. Этапы эргономического проектирования пользовательского интерфейса

Для успешного проектирования цифровых интерактивных продуктов необходимо особое внимание уделять проектированию поведения. При решении данной задачи нельзя рассматривать составляющие ее части отдельно друг от друга. В системе «человек — машина» главным звеном является человек, а система, являясь подчиненным звеном, должна реагировать на его действия, а не наоборот.

В связи с этим появляется новая дисциплина проектирования, которая получает название проектирование опыта взаимодействия и сосредотачивается в основном на проектировании поведения программного продукта.

Проектирование опыта взаимодействия или проектирование взаимодействия — это новая область научно-практической деятельности, которая в последние годы выделяется как самостоятельная дисциплина, сосредоточенная на проектировании поведения пользователя продукта.

Проектирование взаимодействия — это описание возможного поведения пользователя и определение того, как система будет реагировать на его поведение и приспосабливаться к нему.

Потребовалось немало времени, прежде чем разработчики пришли к мысли, что нужно перейти от разработки программного обеспечения, хорошо работающего с точки зрения машины, к созданию программ, хорошо работающих с точки зрения человека.

Проектирование взаимодействия касается не столько эстетических аспектов, сколько понимания потребностей пользователей и принципов их познавательной деятельности. Форма и эстетическая привлекательность продукта должны работать в гармоничной связке при достижении целей пользователей посредством правильно спроектированного поведения продукта.

Для решения этой задачи эффективно применяется инструментарий, включающий методику персонажей, текстовые сценарии взаимодействия, а также проектирование, ориентированное на цели. Весь этот набор можно объединить под названием «проектирование

взаимодействия». При таком подходе к проектированию за отправную точку принимается человек, главная цель — выяснить, чего хочет пользователь.

Проектирование взаимодействия предоставляет описание окончательного варианта продукта, которое содержит предельно ясную и точную информацию о том, кто конкретно будет использовать продукт, каким образом и с какой целью. Имея такое описание, программисты осознают, что именно они создают, руководители могут оценить прогресс в работе программистов, а маркетологи получают понимание источника мотивации покупателя.

Разумеется, проектирование продукта не может концентрироваться только на поведении, необходимо учитывать внешний вид (форму) и информационное наполнение (содержание) разрабатываемого продукта. Поэтому необходимо сочетать подходы различных дисциплин проектирования: проектирование взаимодействия, основное внимание на поведение; информационная архитектура, занимается структурированием содержания; промышленный и графический дизайн, отвечает за форму продуктов и услуг.

Именно грамотная продуманность всех деталей на этих этапах позволит создать армию поклонников вашего продукта. Ярким примером здесь является компания Apple, которая пошла по такому пути и завоевала сердца тысяч и миллионов.

Джесс Гаррет в своей книге «Веб-дизайн. Элементы опыта взаимодействия» представляет проектирование опыта взаимодействия в виде пяти уровней:

- стратегии (или идеи);
- набора возможностей;
- структуры;
- компоновки;
- поверхности (или внешнего вида интерфейса).

Эти пять уровней составляют концептуальную основу для обсуждения связанных с опытом взаимодействия проблем и средств их решения.

Пять наших слоев делятся на две части (рис. 3).

Слева — все, что касается использования web как интерфейса программы. Справа — все, что связано с информационной структурой. Каждый уровень зависит от уровней, расположенных ниже, и диапазон выбора решений ограничен решениями, принятыми на нижних уровнях.

На каждом из уровней размещаются элементы проектирования.

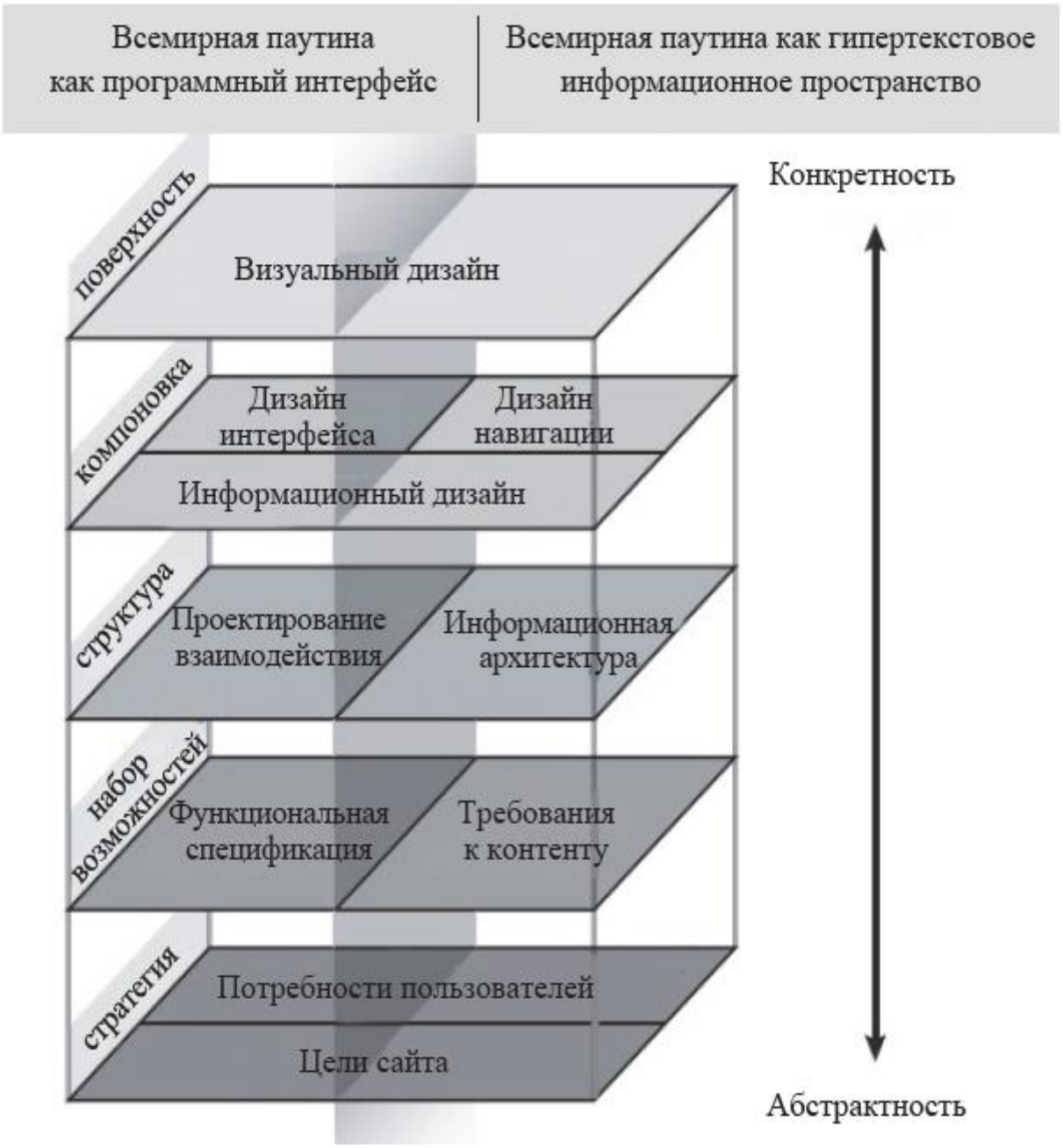

Рис. 3. Элементы проектирования опыта взаимодействия

Уровень стратегии

Цели сайта (Site Objectives) — это бизнес-цели заказчика.

Потребности пользователей (User Needs). Они определяются людьми, которые будут пользоваться нашим проектом. Нужно понимать, чего хочет аудитория и как эти пожелания согласуются с другими ее потребностями.

Уровень набора возможностей

Функциональная спецификация (Functional Specification) — описание функций и возможностей, которые могут быть представлены пользователю.

Требования к контенту (Content Requirements) — это описание различных элементов содержимого, которые необходимо создать.

Уровень структуры

Определяется, как подчинены друг другу разделы, какова иерархия построения информации, как осуществляются переходы между разделами. Если предыдущие уровни визуальные, то уровень структуры логический.

Проектирование взаимодействия (Interaction Design IxD) — выяснение, как система будет вести себя в ответ на действия пользователей.

Информационная архитектура (Information Architecture, IA) организация элементов содержимого в пределах информационного пространства.

Уровень компоновки

Это схематичное разделение страницы на части. Здесь нет информации о внешнем представлении — только о смысле конкретной части страницы, например, блок поиска, блок новостей, блок авторизации. На этом уровне определено, как должны располагаться части страницы.

Информационный дизайн (Information Design) представление информации в таком виде, который облегчает ее восприятие.

Дизайн интерфейса (Interface Design) организация элементов интерфейса, позволяющая пользователям взаимодействовать с функциями системы.

Дизайн навигации (Navigation Design) набор экранных элементов, позволяющих пользователю перемещаться по информационной архитектуре.

Уровень поверхности

Это самый верхний, презентационный уровень. Он представляет собой совокупность шрифтов, таблиц, изображений, цветов, результатов поиска всего, что видит пользователь. На этом уровне определено, как выглядит для посетителя содержимое сайта.

Визуальный дизайн *(Visual Design)* внешний вид конечного продукта. Для получения действительно привлекательных продуктов необходимо, с одной стороны, понимать желания, потребности, мотивации пользователей и контекст, в котором эти пользователи нахо-

дятся, с другой стороны — понимать возможности, требования и ограничения бизнеса, технологии и предметной области. Использование этих знаний в качестве основы всех планов по созданию цифровых продуктов позволяет сделать их полезными, удобными и желанными, а также экономически жизнеспособными и технически осуществимыми.

Три качества: желанность, жизнеспособность и осуществимость — определяют успешный продукт. Ларри Кили (Larry Keeley) выделил современные задачи разработки, которые как раз предполагают реализацию в продукте всех трех качеств и, если хотя бы один из этих трех столпов значительно слабее двух других, продукт вряд ли выдержит испытание временем.

Если говорить об интерфейсе в целом, то сегодня не существует единой методологии их проектирования, поэтому мы будем использовать сочетание различных подходов.

3. Этапы работы над пользовательским интерфейсом

В процессе разработки интерфейса можно выделить пять основных этапов:

1. Сбор функциональных требований (первоначальное проектирование: уровень стратегии, уровень набора возможностей). Данный этап разработки подразумевает под собой сбор, систематизацию и анализ требований к системе. Также анализируются и систематизируются возможные пользовательские системы (персонажи, актеры). Сбор и анализ требований выполняет бизнес-аналитик.

2. Информационная архитектура (уровень структуры). Под информационной архитектурой понимается совокупность методов и приемов структурирования, и организации информации. Другими словами, информационная архитектура занимается принципами систематизации, навигации и оптимизации информации, что позволяет облегчить пользователю работу с данными, а именно их поиск и обработку. За проектирование скелетов пользовательского интерфейса и организацию информационных потоков в приложении отвечает информационный архитектор.

3. Прототипирование пользовательского интерфейса. Это этап разработки, который подразумевает создание прототипов экранов системы. Прототипы позволяют обнаружить проблемы функционального характера будущей системы на раннем этапе и устранить их до того, как проект уйдет в разработку к программистам. Прототипы разрабатываются front-end-разработчиком под руководством UI-дизайнера.

4. Юзабилити-тестирование (Ю-тестирование). К тестированию интерфейса привлекают как конечных пользователей, так и специалистов по функциональному тестированию

ПО. Ю-тестирование позволяет оценить удобство использования продукта. Информационный архитектор проводит Ю-тестирование и анализирует его результаты.

5. Графический дизайн пользовательского интерфейса. Графический облик интерфейса создает UI-дизайнер. На этом этапе интерфейс системы приобретает необходимый законченный вид. Часто заказчик уже имеет брэнд бук или гайдлайн, задача дизайнера — разработать такой дизайн, который бы соответствовал всем требованиям системы, удовлетворял заказчика и сочетался с задумками информационного архитектора.

На данном этапе может понадобиться помощь смежных специалистов: иллюстратора (художника), 3D-моделлера и др.

Разработанный по всем правилам пользовательский интерфейс значительно повышает эффективность ресурса и дает конкурентные преимущества.

4. Первоначальное проектирование: сбор функциональных требований

Важность этого этапа трудно переоценить. На нем закладываются основные концепции системы, влияющие абсолютно на все показатели качества ее интерфейса. Структурные проблемы практически не могут быть обнаружены и решены на остальных этапах (для их обнаружения нужно слишком много везения, а для исправления — денег). Это значит, что чем больше внимания будет уделено проектированию, тем выше будет общее качество.

Прежде чем начать любое проектирование, необходимо выполнить исследование предметной области, которое позволяет проектировщикам лучше понять цели бизнеса, атрибуты бренда и технические ограничения. Необходимо посмотреть, как организована работа на аналогичных проектах, проанализировать сильные и слабые стороны этих работ, насколько удобно все сделано, насколько грамотно подана информация.

Основной задачей первоначального проектирования является определение необходимой функциональности системы.

Современная наука выдвинута два основных способа определения функциональности, а именно анализ целей и анализ действий пользователей.

Анализ целей пользователей

Хорошо сформулированная цель должна быть:

- понятной. Избегайте использования узкоспециализированной терминологии;
- ясной. Избегайте туманных формулировок; подбирайте выражения, которые были бы уместными при определении приоритетов требований;

• измеримой. Используйте конкретные утверждения, которые можно проверить независимо, чтобы определить степень успешности проекта.

Идея, лежащая в основе, — *«людям не нужны инструменты сами по себе, нужны лишь результаты их работы».* Никому не нужен текстовый процессор, нужна возможность с удобством писать тексты.

Очень важно различать задачи и цели. Программисты часто путают их. Они обычно занимаются проектированием, ориентированным на задачи. Программисты начинают проектирование с вопроса: *«Каковы задачи?».* Компьютерное программирование, если добраться до сути, — это создание подробных пошаговых описаний процедур. Процедура есть рецепт решения задачи. Хорошие программисты имеют «процедурный» взгляд на вещи, взгляд, ориентированный на решение задач.

Такой подход дает возможность сделать работу, но не позволяет даже приблизиться к наилучшему решению, а также совершенно не удовлетворяет пользователя.

Цель — это конечное состояние, тогда как *задача* — переходный процесс, необходимый для достижения цели. *Цель* — стабильная сущность. *Задачи* — преходящи.

Вопрос «Каковы *цели* пользователя?» позволяет создавать более качественный и уместный дизайн. Для достижения целей пользователя, конечно, необходимо решать и задачи, однако существуют разные акценты и различные последовательности выполнения задач.

В случаях, когда для решения поставленных проблем проектировщики взаимодействия анализируют цели, они обычно находят совсем иные, более подходящие решения.

Рассмотрим цели более подробно. Существуют следующие виды целей: личные, практические, корпоративные, ложные.

Личные цели:

1. не чувствовать себя глупо;
2. не совершать ошибок;
3. выполнить адекватный объем работы;
4. развлечься (или хотя бы не страдать от скуки).

Личные цели всегда истинны и действительны в определенных рамках для всех людей. Они всегда предшествуют всем другим целям, хотя очень редко становятся предметом обсуждения — как раз потому, что являются личными.

Любая система, идущая вразрез с личными целями, в конечном итоге обречена на неудачу, независимо от того, насколько качественно позволяет достигать целей иного рода.

Корпоративные цели. У каждого делового предприятия свои требования к программному обеспечению, и уровень этих требований достаточно высок:

1. увеличить прибыль;

2. увеличить рыночную долю;
3. победить конкурентов;
4. нанять больше сотрудников;
5. предложить новые продукты и услуги;
6. выпустить акции компании в свободное обращение.

Цель «увеличить прибыль» является преобладающей для совета директоров или держателей акций. Эти цели необходимы для эффективной работы, но сами по себе не мотивирующие. С точки зрения корпорации это все важные цели, однако работу выполняет не корпорация, а люди, а для людей важнее цели личные.

Сущность качественного проектирования взаимодействия состоит в том, чтобы позволить пользователям достигать практических целей, не отказываясь от целей личных.

Программа, которая не позволяет достичь какой-либо корпоративной или личной цели, потерпит неудачу.

Практические цели:

- удовлетворять требованиям клиента;
- сохранять информацию о заказах клиента;
- создавать математические модели бизнеса.

Практическая цель удовлетворения требований клиента соединяет корпоративную цель (более высокие прибыли) с личной целью пользователя (работать продуктивно).

Обычно программисты создают программное обеспечение, которое замечательно помогает достигать практических целей, но совершенно неспособно удовлетворить пользователей. Разумеется, чтобы удовлетворить цели бизнеса, вы должны встроить в программу определенные возможности. Однако, если пользователь не может достичь личных целей, то он не способен эффективно достигать целей компании. Непреложный факт: счастливые и довольные сотрудники наиболее эффективны.

С другой стороны, если программа игнорирует практические цели и служит только целям пользователя, это означает, что вы спроектировали компьютерную игру.

Ложные цепи:

1. экономия памяти;
2. уменьшение потребности в клавиатурном вводе;
3. поддержка работы в браузере;
4. простота в освоении;
5. обеспечение целостности данных;
6. ускорение ввода данных;
7. увеличение скорости исполнения программы;

8. применение супертехнологии или супервозможностей;
9. улучшение внешнего вида;
10. сохранение единообразия интерфейса на различных платформах.

Повседневные продукты, основанные на программном обеспечении, создаются на основе ложных целей. Многие из этих целей облегчают задачу создания программ, в чем и заключается цель программиста, и потому их приоритет повышается во вред конечному пользователю. Другие ложные цели связаны с задачами, возможностями и инструментами. Все это средства достижения результатов, но еще не результаты, тогда как цели всегда являются результатами.

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.

Анализ действий пользователей

Достижение почти всех целей требует от пользователей совершения определенных действий. Разумеется, эти действия могут различаться при разных способах достижения.

Единственным же способом проверить, нужна функция или нет, является наблюдение за пользователями и анализ их действий.

Поскольку на этом этапе мы узнаем, какая именно функциональность нужна для каждого варианта, можно избрать верный путь по правилу *«чем меньше действий требуется от пользователя, тем лучше».* Не стоит забывать и про другое правило: *чем меньше функций, тем легче их сделать*.

Создание пользовательских сценариев

Цель 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 очень удобно различать по именам участвующих в них вымышленных персонажей.

Алан Купер предложил эффективный инструмент для анализа действий пользователя: это точное описание пользователя продукта и его целей. Для этого мы выдумываем несуществующих пользователей и проектируем для них.

Таких несуществующих пользователей называют *персонажами (personas),* и они представляют собой необходимую базу качественного проектирования взаимодействия.

Краткая справка: «персонаж» (франц. *personnage,* от лат. *persona* — личность, лицо) — действующее лицо пьесы (спектакля), сценария (кинофильма), романа и других художественных произведений.

Персонажи не реальные люди, но они представляют реальных людей в процессе проектирования. Будучи воображаемыми, они тем не менее определяются достаточно жестко и точно. На практике действительно выдумываются их имена и личные сведения.

Персонажи определяются своими целями. Цели же, разумеется, определяются персонажами.

Персонаж должен быть конкретным**.** Чем более конкретными мы делаем персонажи, тем более эффективными инструментами проектирования они становятся. Для этого мы выбираем ему имя.

Персонаж должен быть воображаемым. Описание персонажа должно быть подробным, а не идеальным, т. е. важнее определить персонаж как можно подробнее и конкретнее, чем создать абсолютно правильный персонаж.

Изобретенные персонажи уникальны для каждого проекта.

Точно определенный персонаж дает нам:

- определенность относительно уровня владения пользователя компьютером, поэтому мы перестаем терзаться загадкой, для кого проектировать: для дилетанта или специалиста; реалистичный взгляд на уровень подготовленности пользователей;
- возможность объяснять наши решения в области проектирования. Это как прожектор, высвечивающий для разработчиков, маркетологов, руководителей очевидную правильность наших решений по проектированию.

Программистам свойствен математический подход, и они естественным образом не склонны рассматривать отдельных пользователей, предпочитая обобщение. Ключ к успеху в том, чтобы заставить программистов поверить в существование и реальность созданных персонажей. Примерив на персонаже продукт или задачу, вы сразу можете понять, удастся ли вам его удовлетворить.

Можно выделить шесть типов персонажей, которые назначаются обычно в таком порядке:

1. *ключевой* задает основную цель в проектировании интерфейса, выбирается методом исключения: цели каждого персонажа рассматриваются в сравнении с целями остальных. Если не очевидно, какой из персонажей является ключевым, это может означать одно из двух: или продукту требуется несколько интерфейсов, каждый из которых предназначен для своего ключевого персонажа (так часто бывает в корпоративных и технических продуктах), или же объем его функциональности слишком широк;

2. *второстепенный* в основном оказывается доволен интерфейсом ключевого персонажа, но имеет дополнительные потребности, которые можно включить в продукт, не нарушая его способности служить ключевому персонажу;

3. *дополнительный* — пользовательский персонаж, не являющийся ни ключевыми, ни второстепенным. Их нужды обычно полностью представлены сочетанием нужд ключевого и второстепенных персонажей и удовлетворяются одним из ключевых интерфейсов;

4. *покупатель* – персонаж, отражающий потребности покупателей, а не конечных пользователей. Обычно персонажи покупателей используются в качестве второстепенных персонажей. Однако в некоторых корпоративных средах кто-то из таких персонажей может оказаться ключевым, если ему предназначается собственный административный интерфейс;

5. *обслуживаемый* — не является пользователем продукта, однако его непосредственно затрагивает применение продукта. Обслуживаемые персонажи — это способ отслеживать социальные и физические воздействия второго порядка, оказываемые продуктом. Эти персонажи используются так же, как второстепенные;

6. *отвергаемый* – используется, чтобы демонстрировать заинтересованным лицам и участникам разработки, что существуют пользователи, для которых продукт не предназначен.

Подбор персонажей

Каждый проект получает собственный набор персонажей в количестве от трех до двенадцати. Мы проектируем не для каждого из них, но все персонажи полезны для выражения пользовательской аудитории. В каждом наборе персонажей есть хотя бы один ключевой персонаж. Эта личность находится в фокусе процесса проектирования.

Персонажи и цели неразделимы, они как разные стороны одной медали. Персонаж существует потому, что у него есть цели, а цели существуют, чтобы придавать смысл персонажу.

Следует отметить, что набор характеристик, подробно описывающий пользователя, зависит от предметной области и контекста решаемых им задач. Наиболее общий шаблон профиля содержит в себе следующие разделы:

- социальные характеристики (фотография, имя, возраст, место жительства, род занятий и биография);
- навыки и умения работы с компьютером;
- мотивационно-целевая среда;
- рабочая среда;

• особенности взаимодействия с компьютером (специфические требования пользователей, необходимые информационные технологии и др.).

Фотография. Выбирая фотографию, следите за тем, чтобы изображение не казалось постановочным или «глянцевым». Персонаж, снятый в естественной обстановке, создает более яркий и достоверный образ.

Имя. С внешним обликом необходимо связать имя. «Нинель» не только лучше звучит, чем «Блондинка, за 30, работает с детьми», но и проще запоминается и связывается с конкретным персонажем. Не следует использовать имена коллег или заказчиков, хотя такая идея и выглядит соблазнительно.

Возраст. Между моделями поведения 21-летней студентки и 34-летней домохозяйки существуют серьезные различия!

Место жительства. В Италии, например, в разных областях страны говорят на разных диалектах. В Республике Беларусь стоимость потребительской корзины жителя Минска, скорее всего, будет отличаться от стоимости потребительской корзины жителя Крупок.

Род занятий. Представление о том, чем ваш персонаж зарабатывает себе на жизнь, поможет вам лучше его понять, так как вы сможете опираться на типичные события и ситуации его повседневной жизни. Персонаж, работающий в поликлинике, ежедневно контактирует со многими людьми, тогда как в жизни оператора подъемного крана вряд ли есть избыток общения.

Биография — убедительная история, которая делает персонаж реалистичным. Она должна быть достоверной, так что наделение персонажа чертами реальных людей — вполне допустимый прием. Сделайте все необходимое, чтобы персонаж выглядел достоверно и как можно более осмысленно для проекта, над которым вы работаете.

Профили пользователей могут по необходимости расширяться за счет добавления других (значимых с точки зрения проектировщика) характеристик пользователей.

При создании персонажей необходимо предоставить достаточно информации, чтобы увлечь людей и заставить их почувствовать человека, описание которого они читают.

Дополнительный контент. Базовые подразделы — своего рода «необходимый минимум» для всех создаваемых вами персонажей. В большинстве случаев вам придется добавить к ним те или иные дополнительные элементы. Ценность ваших персонажей можно повысить, если дополнить их описание следующими подразделами.

1. Образование. Человек с аттестатом средней школы может серьезно отличаться своим покупательским поведением и восприятием бренда от человека со степенью магистра; таким образом, эта информация способна повлиять на то, как будет восприниматься ваш персонаж.

2. Заработная плата. Эта информация способна привести к значимым открытиям, если вы ориентируетесь на определенные уровни состоятельности.

3. Интернет-активность. Учитывая, что сейчас многие проекты имеют интернет-составляющую, при подготовке этого элемента следует руководствоваться здравым смыслом.

4. Ключевая точка взаимодействия с заказчиком, брендом или проектом. Персонаж узнал о них от своих знакомых, по телевизору или радио, прочитал в интернет-обзоре, на форуме или во всплывающем рекламном окне? Используйте статистические данные для прояснения этого момента и включите результат в описание персонажа — это поможет заложить основу для привлечения пользователей к проекту.

5. Техническая подготовка. На каком компьютере работает ваш персонаж — на PC или на Mac? У него есть собственный компьютер? Использует ли он системы мгновенного обмена сообщениями, Flickr, ведет ли блог? Насколько уверенно чувствует себя при этом? Поможет ли ему очень простое решение, рассчитанное на новичка? Есть ли у него MP3-плеер или другое портативное устройство? Использует ли он DVR, AppleTV или другие устройства для просмотра телепрограмм? Этот список может быть очень длинным. В зависимости от заказчика, бренда или проекта такие мелочи могут сыграть важную роль.

6. Уровень социального комфорта. Учитывая бурный рост сетевых сообществ и социальных сетей, точное описание степени участия персонажа в сетевом пространстве может многое рассказать о нем. Есть ли у него учетная запись Twitter? Если есть, то сколько у него подписчиков? Насколько персонаж активен? Является ли он лидером? Использует ли MySpace, Facebook, LinkedIn, другие агрегаторы или сетевые сообщества?

Разработка сценариев

На данном этапе уточняется, какими должны быть информация и функциональные возможности интерфейса, чтобы пользователь дошел до целевого действия.

Сценарий — это описание действий, выполняемых пользователем в рамках решения конкретной задачи на пути достижения его цели. Очевидно, что достигнуть некоторой цели можно, решая ряд задач. Каждую из них пользователь может решать несколькими способами, следовательно, должно быть сформировано несколько сценариев. Чем больше их будет, тем ниже вероятность того, что некоторые ключевые объекты и операции будут упущены.

Эффективность сценария определяется в большей степени его охватом, чем глубиной. Иначе говоря, важнее, чтобы сценарий описывал процесс от начала до конца, чем чтобы он описывал каждый шаг в исчерпывающих подробностях.

Важно развивать лишь те сценарии, которые позволяют продвигаться вперед в процессе

проектирования. Достаточно разработать лишь два вида сценариев, хотя сценариев каждого вида может быть и несколько.

На различных этапах целеориентированного проектирования используются разные типы сценариев, основанных на персонажах, причем на каждой последующей стадии особенностям интерфейса уделяется больше внимания, чем на предыдущей.

Первый тип сценариев контекстные сценарии — используется для высокоуровневого рассмотрения того, как продукт может наилучшим образом послужить потребностям персонажей. (Раньше эти сценарии называли «день из жизни», но в конечном итоге сочли, что этот термин является слишком общим).

Контекстные сценарии создаются на стадии первоначального проектирования, пишутся с точки зрения персонажа и сосредоточены на человеческих действиях, впечатлениях и желаниях. При разработке именно этого вида сценариев проектировщик располагает наибольшей свободой в представлении идеального опыта пользователя.

После того как команда проектировщиков определила функциональные и информационные элементы, а также создала общую инфраструктуру, необходимо пересмотреть контекстный сценарий. В результате добавления к нему более подробных описаний взаимодействия пользователя с продуктом и применения проектного лексикона он становится сценарием ключевого пути.

Сценарии ключевого пути фокусируются на наиболее важных моментах взаимодействия, не теряя из виду того, как персонаж пользуется продуктом при достижения своих целей. По мере уточнения образа продукта эти сценарии параллельно с проектированием проходят итерационную доработку.

В ходе всего процесса команда проектировщиков применяет проверочные сценарии для тестирования проектных решений в различных ситуациях. Как правило, эти сценарии менее подробны и обычно принимают форму набора вопросов «а что, если?..», касающихся предложенных решений.

На основе выявленных сценариев работы осуществляется разработка структуры экранов, т. е. определяется количество экранов, функциональность каждого из них, навигационные связи между ними, формируется структура меню и других навигационных элементов.

Сценарии представляют примеры использования как отправную точку для проектирования, а также закладывают основу для юзабилити-тестирования. Сценарии являются реалистичными и детализированными описаниями действий пользователей, но в них не должно быть ссылок на применение каких-либо элементов пользовательского интерфейса.

Основная сложность при использовании этого метода связана с осознанной необходимостью разработки такого количества сценариев, которое покрывало бы наибольшее количество различных ситуаций, а не только самых типичных или, например, интересных разработчикам. Наряду с последовательными, в список стоит включить и нелинейные сценарии, которые будут использованы при тестировании. В дальнейшем, для оценки разрабатываемой системы, должен использоваться полный набор сформированных сценариев. Для создания сценария проанализируйте следующие вопросы.

1. Кто является главным пользователем в этом сценарии?
2. Посещал ли выбранный пользователь этот сайт ранее (работал ли он ранее с данным программным продуктом)?
3. Какие срочные потребности привели пользователя на сайт (либо заставили обратиться к данной программе)?

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

Елизавета Бронеславовна запускает почтовую программу. Она включает процесс скачивания новой почты. Получив почту, она читает все сообщения, затем часть их удаляет, а на одно сообщение отвечает. После чего выключает почтовую программу.

Еремей Карлович 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;

Пришло новое сообщение, и Эмма Валерьевна восприняла соответствующий индикатор. Она делает активным окно почтовой программы и открывает полученное сообщение. Читает его, после чего перемещает в другую папку. Затем переключается на другую задачу.

Дело в том, что на таких сценариях очень хорошо заметны ненужные шаги, например, в третьем сценарии гипотетическая Эмма Валерьевна 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.

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.

5. Проектирование общей структуры системы

После выделения нескольких профилей пользователей и определения целей, и задач, стоящих перед ними, а также пользовательских сценариев переходим к проектированию общей структуры системы, т. е. необходимо выделить отдельные функциональные блоки и определить, как именно эти блоки связываются между собой. Под отдельным функциональным блоком будем понимать функцию / группу функций, связанных по назначению или области применения в случае программы, и группу функций / фрагментов информационного наполнения в случае сайта.

Типичная структура сайта (слева) и типичная структура программы представлены на рис. 4. Если сайты обычно разветвлены, в том смысле, что функции обычно размещаются в отдельных экранах, то программы обычно имеют только один изменяющийся экран, в котором и вызываются почти все функции.

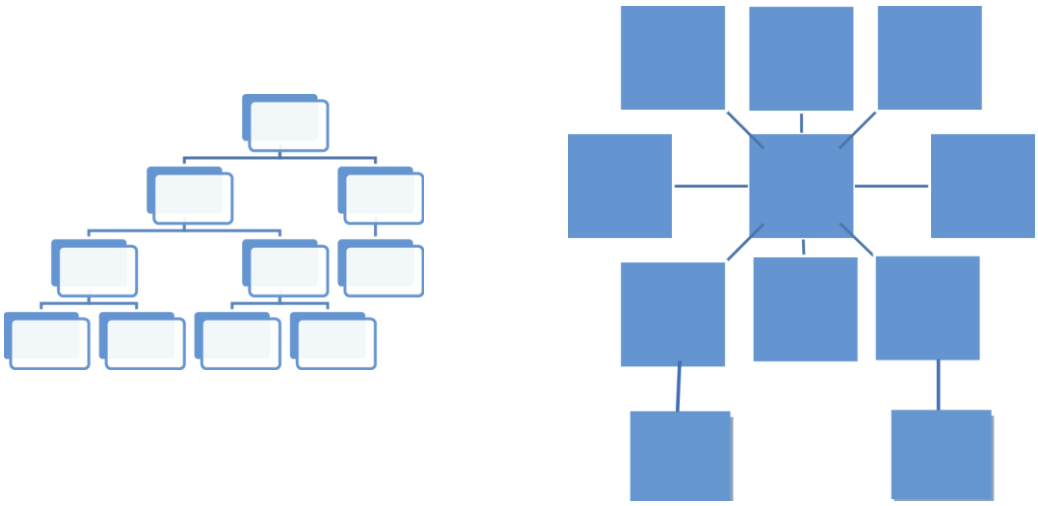

Рис. 4. Типичная структура сайта (слева) и типичная структура программы (справа)

Проектирование общей структуры состоит из двух параллельно происходящих процессов: выделения независимых блоков и определения связи между ними. Если проектируется сайт, в завершении необходимо также создать схему навигации.

Выделение независимых блоков

Выделение независимых блоков выполняется на основе выявленного с помощью пользовательских сценариев перечня отдельных функций. В приложении функция представлена функциональным блоком с соответствующей экранной формой (формами). Как правило,

несколько функций объединяются в один функциональный блок. Избегайте 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.

Определение связи между блоками

Существует три основных вида смысловой связи между блоками: логическая связь, связь по представлению пользователей, процессуальная.

Логическая связь определяет взаимодействие между фрагментами системы с точки зрения разработчика (суперпользователя). Данная связь очень существенно влияет на навигацию в пределах системы (особенно, когда система многооконная). Чтобы не перегружать интерфейс, стоит избегать блоков, связанных с большим количеством других, — оптимальным числом связей является число три.

Связь по представлению пользователей. В информационных системах, когда необходимо гарантировать, что пользователь найдет всю нужную ему информацию, нужно устанавливать связи между блоками, основываясь не только на точке зрения разработчика, но и на представлениях пользователей.

Например, нужно как-то классифицировать съедобные растения. Помидор, который почти все считают овощем, на самом деле ягода. Не менее тяжело признать ягодой арбуз. Это значит, что классификация, приемлемая для ботаника, не будет работать для всех остальных, причем обратное не менее справедливо.

Процессуальная связь описывает пусть не вполне логичное, но естественное для имеющегося процесса взаимодействие: например, логика напрямую не командует людям сначала приготовить обед, а потом съесть его, но обычно получается именно так.

Жестко заданная процессуальная связь позволяет также уменьшить количество ошибок, поскольку от пользователя при ней не требуется спрашивать себя: «Не забыл ли я чего?».

Замечательным примером жестко заданной процессуальной связи является устройство мастеров (wizards), при котором пользователя заставляют нажимать кнопку «Далее».

Существует любопытная закономерность: чем эстетически привлекательней выглядит схема (без учета цветового кодирования и интересных шрифтов), тем она эффективней. Всегда надо стараться сделать схему возможно более стройной и ясной.

Навигационная система

На основе разработанной структуры экранов на этом этапе выбирается наиболее адекватная навигационная система и разрабатывается ее детальный интерфейс. Навигационная система показывает механизм распределения функций и задач между экранами и объектами страниц.

Процесс навигации должен быть спроектирован таким образом, чтобы помочь пользователям определить, где они находятся, где они находились и куда они могут переместиться в дальнейшем. Для этой работы трудно дать какие-либо конкретные рекомендации, поскольку очень многое зависит от проектируемой системы. Например, рассмотрим систему для сайта новостей.

Незарегистрированный читатель нашел новость — открыл новость — прочитал — поделился с друзьями — прокомментировал — добавил в избранное — вернулся на главную.

Для того, чтобы представить эту схему в графическом виде, нужно каждому этапу присвоить емкое название и придумать блоки, которыми должен быть дополнен каждый этап. Важно не перегрузить пользователя информацией и одновременно включить в схему важные функции. Для этого надо составить полный список всего, что пользователь должен видеть на экране и разделить эту информацию на четыре категории: обязан быть; должен быть; может быть; мог бы быть.

К обязательным элементам нужно отнести функции, решающие задачи пользователей (новости). К должным — функции, упрощающие достижение этих задач (категории, поиск). К остальным относятся функции, дополняющие решение задач или осуществляющие реализацию других задач (избранное, перепост, комментарий и т. д.).

Навигационные связи между отдельными функциональными блоками отображаются на схеме навигационной системы. Возможности навигации в приложении передаются через навигационные элементы.

Основным навигационным элементом приложения является главное меню. Формирование меню начинается с анализа функций приложения. Для этого в рамках каждой из них выделяют отдельные элементы: операции, выполняемые пользователями, и объекты, над которыми осуществляются эти операции. Следовательно, известно, какие функциональные блоки должны позволять пользователю осуществлять определенные операции над определенными объектами. Выделение операций и объектов удобно проводить на основе пользовательских сценариев и функционала приложения. Выделенные элементы группируются в общие разделы главного меню. Группировка отдельных элементов происходит в соответствии с представлениями об их логической связи. Таким образом, главное меню может

иметь каскадные меню, выпадающие при выборе какого-либо раздела. Каскадное меню ставит в соответствие первичному разделу список подразделов.

Интуитивно понятная навигация и положительные ощущения пользователя обусловлены не чем иным, как правильным расположением и представлением информации.

После создания навигационной карты следует еще раз вернуться к портрету своей целевой аудитории и ролям пользователей в приложении. На примерах этих ролей следует снова пройти все этапы навигации внутри приложения, уделив особое внимание функционалу. В конечном итоге, целью пользования любым приложением является получение информации.

Виды структуры Web-сайта

Структура сайта разделяется на внутреннюю и внешнюю. При этом внутренняя структура зачастую значительно влияет на внешнюю.

Внутренняя структура сайта. К ней относятся логические связи между различными страницами ресурса. В данной части необходимо продумать, как пользователь сможет максимально быстро получить доступ к нужной информации. К примеру, позаботиться, чтобы человеку потребовалось не более трех кликов для перехода ко всем важным разделам или интересным страницам. Также иногда внутренней структурой называют особенности размещения директорий и ресурсов на сервере.

Внешняя структура сайта. Она полностью повторяет навигацию ресурса и используется для того, чтобы упростить «путешествие» посетителей по страницам. Благодаря ей человек может получить доступ к основному функционалу сайта с любой страницы.

При этом внешняя структура анализируется поисковыми системами и может повлиять на позицию вашего сайта в выдаче.

Типовые структуры сайтов

Различают четыре основные типовые структуры сайтов. В чистом виде каждая из них используется достаточно редко. Чтобы сделать сайт максимально удобным для пользователя и не навредить его функционалу, следует грамотно комбинировать различные виды структур Web-сайтов.

Линейная структура сайта. Данный вид сайта является наиболее простым. Подобная структура последовательна, в ней каждая из страниц ведет на предыдущую и следующую страницы ресурса. В данном случае навигация очень проста и осуществляется с помощью 2-3 ссылок, использование такого сайта чем-то похоже на перелистывание страниц книги.

Решетчатая структура сайта. Разработка структуры веб-сайта подобного типа также не

отличается особой сложностью, но она более удобна для использования, чем линейный вариант. В данном случае каждая страница ресурса связана с двумя или тремя страницами одновременно. Обычно каждая из них имеет связи с одной или двумя страницами разделов одного уровня, а также с одной страницей — подразделом. Здесь уже должна прослеживаться иерархическая структура информации на ресурсе.

Иерархическая структура сайта. Основным элементом иерархической структуры является главная страница сайта. Ссылки с нее ведут на разделы второго уровня, а на страницах второго уровня размещены ссылки на материалы/разделы третьего уровня и т. д. В данном случае пользователь, перейдя на главную страницу сайта, должен обязательно посетить страницу определенного раздела, чтобы добраться до подраздела. Зачастую подобная структура Web-сайта применяется для каталогов товаров.

Паутинообразная структура сайта. Она предполагает связь страниц «все со всеми», то есть пользователь может попасть с любой страницы ресурса на любую другую страницу, минуя все разделы. Разработка структуры веб-сайта подобного типа достаточно сложна, особенно если на ресурсе расположено много страниц. На практике пользователь просто «потеряется» в огромном количестве ссылок. Стоит отметить, что данная структура Web-сайта повторяет в малом масштабе структуру всемирной сети Интернет.

Гибридная структура сайта. Для повышения удобства использования сайта разрабатывают гибридную структуру. Выше описанные структуры Web-сайтов практически не пригодны для использования в чистом виде. Они являются конструктором, из которого каждый может взять необходимые ему блоки для реализации своего проекта. К примеру, для каталога товаров интернет-магазина лучше всего использовать иерархическую структуру. При этом на самой странице товара можно расположить ссылки на сопутствующие товары, чтобы пользователь долго их не искал, а это уже элемент паутинообразной структуры.

Главное требование к разработке структуры — это логичность и простота. Пользователь должен с легкостью находить нужные ему материалы. Необходимо отметить, что любой Web-сайт не может постоянно поддерживать одну и ту же структуру. Несмотря на то, что разделы и каталоги будут оставаться неизменными, при размещении новых материалов и статей будет возникать внутренняя перелинковка, которая внесет свои корректировки.

Сайты отличаются друг от друга по размеру и предназначению. Интернет-магазины имеют сотни или тысячи страниц, блоги могут обойтись несколькими десятками страниц, а сайтам-визиткам достаточно одной-двух. Поэтому нельзя придумать универсальный шаблон структуры, который подойдет каждому Web- сайту.

Создание глоссария

Еще в процессе проектирования полезно зафиксировать все используемые в системе понятия. Для этого нужно просмотреть 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 необходимо добавить определения всех концепций системы (например, книга или изображение). Затем этот список нужно улучшить.

- уменьшить длину всех получившихся элементов;
- показать этот список любому потенциальному пользователю системы и спросить его, как он понимает каждый элемент. Если текст какого-то элемента воспринимается неправильно, его нужно заменить;
- 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);
- убедиться, что на всех командных кнопках стоят глаголы-инфинитивы (создать, отправить, сохранить).

Нужно стараться не менять список в будущем.

После завершения проектирования структуры интернет-ресурса можно будет перейти к процессу создания страниц, разработки их дизайна и наполнения контентом.»

Результатом выполнения первых двух этапов проектирования ПИ проектирования является полная функциональная схема, описывающая все взаимодействие пользователя с системой, т.е. уже решено, сколько экранов (страниц) будет в системе и что должно находиться на каждом экране.