Персонификация

Этот метод подразумевает составление детализированных типовых профилей потенциальных пользоватей, относящихся к разным группам. Анализ профилей позволяет смоделировать такие поведенческие аспекты, как цели, желания, потребности, предпочтения и ожидания пользователей. Это будет полезным при принятии решений, связанных с возможностями продукта, их визуальным представлением и способами интерактивного взаимодействия.

Анализ контекста

Анализ контекста использования состоит в сборе всей доступной информации о том, что именно делают пользователи в процессе выполнения конкретной задачи и в каком окружении они это делают. Это позволяет направить разработку интерфейса так, чтобы он наиболее полно соответствовал порядку работы пользователей с компонентами системы. Результаты анализа являются основой для составления *сценариев использования* (Use Cases).

Сценарии использования (Use cases)

Сценарии описывают поведение пользователей при решении производственных задач в определенном контексте. Они представляют примеры использования как отправную точку для проектирования, а также закладывают основу для юзабилити-тестирования.

Преимуществами использования сценариев является то, что они позволяют:

* моделировать поведение предполагаемых пользователей, их задачи и окружение;
* исследовать вопросы юзабилити на самых ранних этапах проектирования;
* определять цели пользователей и вероятное время, затрачиваемое ими для достижения этих целей;
* обойтись минимальными ресурсами;
* использовать сценарии для дальнейших оценочных исследований;
* уменьшить необходимость экспертизы человеческого фактора.

Алгоритм разработки пользовательских сценариев может быть представлен следующим образом:

1. Определение общего контекста, выделение потенциальных пользователей и их задач в этом контексте.
2. Функциональная декомпозиция пользовательских задач на последовательности операций, необходимых для их решения.
3. Разделение операций на те, которые должны выполняться пользователями и те, которые компьютером.
4. Непосредственное формирование сценариев в виде последовательности операций. При этом не следует выделять, что для решения определенных задач используются какие-то особенности продукта.
5. Дополнение сценариев оценками времени и критериями завершенности.

**Примечание:** Сценарии являются реалистичными и детализированными описаниям действий пользователей, но в них не должно быть ссылок на применение каких-либо элементов пользовательского интерфейса (см. [примеры](http://academy.kgtk.ru/it3/user-interfaces/use-cases-examples.html)).

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.

Сортировка карточек

Это простой, надежный и недорогой метод изучения пользователей, применяемый для деления информации на группы. Результаты сортировки (полученные группы) могут использоваться для структуризации приложения и, как следствие, формирования навигационной схемы (например, определение структуры меню веб-сайта).

Суть метода сортировки карточек сводится к следующему:

1. *Формирование списка материалов и тематик.* Для этого используются различные источники, начиная от материалов, используемых в имеющемся приложении (или в конкурирующих разработках) и вплоть до планируемых в будущих версиях. Включение будущих материалов, которые не предусмотрены в текущей разработке, позволит в дальнейшем сократить затраты, поскольку возможность расширения функциональности и представляемой информации уже будет спроектирована.
2. *Подбор участников.* Сортировка карточек может выполняться индивидуально или в группе. Для индивидуального тестирования потребуется с десяток добровольцев. Для группового тестирования рекомендуется сформировать не менее пяти групп по три человека в каждой. В обоих случаях главное то, что участники тестирования должны быть наиболее типичными представителями целевой аудитории.
3. *Подготовка карточек.* Тем или иным способом ранее отобранные материалы наносят на отдельные бумажные карточки. Подписи на карточках должны быть достаточно короткими, чтобы участники могли их быстро прочитать и в то же время достаточно подробными, чтобы участники могли понять о чем идет речь. Рекомендуется оставить несколько пустых карточек, куда участники тестирования смогут вписать свои предложения. Все карточки, в т.ч. и пустые, снабжаются уникальным идентификатором.
4. *Выполнение теста.* Перед началом теста карточки перемешивают, чистые карточки помещают рядом. Участники теста по одному (или по группам) заходят в комнату и раскладываю карточки так, как считают нужным, при необходимости — записывают свое видение в пустые карточки. Наблюдатель, постоянно присутствующий в комнате, фиксирует результаты сортировки, карточки снова перемешивают и приглашают следующего участника (группу).
5. *Анализ результатов.* Результаты тестов сводят в единую таблицу и уже по ней выявляют те самые пользовательские предпочтения, ради чего все это и затевалось. Здесь нет каких-либо точных инструкций, поскольку любой анализ есть «нечто среднее между магией и наукой».

[Донна Спенсер](http://rosenfeldmedia.com/experts/donna-spencer/), автор книги «Card Sorting», использует этот метод в проектировании веб-сайтов и приводит рекомендации, когда сортировка карточек дает хорошие результаты, а когда — трудные для анализа (табл. 1).

Табл. 1. Применение метода сортировки карточек

|  |  |  |
| --- | --- | --- |
|  | Просто | Трудно |
| Размеры сайта | Малый | Большой |
| Тип материалов | Однородные (напр., каталог товаров, список услуг, блог и т.д.) | Разнородные (напр., портал, правительственный сайт и т.п.) |
| Сложность материалов | Участники разбираются в содержании большинства материалов | Материалы требуют специфических или специальных знаний |

Метод сортировки карточек имеет [несколько разновидностей](http://academy.kgtk.ru/it3/user-interfaces/usability-methods.html#essay), информацию о которых вам предлагается отыскать самостоятельно.

**Анализ конкурентов** — простой, недорогой и эффективный метод, позволяющий выявить сильные и слабые стороны программных продуктов или сервисов, аналогичных проектируемому, но уже имеющихся на рынке. Небольшое время, потраченное на ознакомление с несколькими наиболее популярными аналогами и представление использованных в них способов решения типичных задач на обсуждение заинтересованным сторонам, исключает необходимость «изобретать велосипед». В ходе обсуждения преимущества и недостатки сторонних разработок анализируются, а результаты фиксируются в виде перечня вопросов, которые предстоит решить, чтобы обойти конкурентов. Также результатом применения этого метода может являться список возможностей, которые, возможно, потребуется включить в новый продукт ([подробней о методе](http://www.usabilitynet.org/tools/competitoranalysis.htm)).

Диаграммы близости

Оригинальное название этого метода — *affinity diagramming* — можно перевести как построение диаграммы тематического сходства/близости. Метод основан на [сортировке карточек](http://academy.kgtk.ru/it3/user-interfaces/usability-methods.html#cardSorting), но выполняется иначе: группировкой элементов занимаются представители разработчика и эксперты со стороны заказчика в ходе совместного обсуждения. Участникам представляется возможность реструктурировать элементы и/или группы, добавлять новые и удалять не нужные.

Мозговой штурм

Широко используемый экспертный метод оперативного решения задач. Поиск решения выполняется в три этапа:

1. *Постановка задачи*. В ходе этого этапа проблема, подлежащая решению, должна быть четко сформулирована.
2. *Генерация идей*. Основной этап, на котором от участников требуется быстро предлагать различные, возможно даже абсурдные идеи решения задачи. На этом этапе исключены какие-либо оценки предлагаемых вариантов, поскольку здесь главное — их количество.
3. *Группировка, оценка и отбор идей*. Каждая из предложенных идей обсуждается и принимается решение о возможности ее дальнейшего использования.

Очень часто метод мозгового штурма применяют «внутри» других методов, например, в ходе проектирования структуры приложения методом Диаграммы близости.

Фокус-группы

Фокус-группа — это неформальное собрание пользователей, у которых запрашивается мнение по определенной теме. Цель в том, чтобы выявить чувства, восприятие, общее отношение и идеи участников обсуждения применительно к обсуждаемому вопросу. Метод фокус-групп применяется, в первую очередь, для сбора информации, но не для ее оценки, поэтому важно так начать дискуссию, чтобы пользователи перешли к активному обсуждению. Иначе, можно получить ответы не столько выражающие мнение участников, сколько ожидаемые организаторами. Фокус-группы часто применяются для тестирования ранее внедренной или внедряемой системы. Положительным аспектом этого метода является то, что в ходе обмена мнениями пользователи обучают друг друга.

Дневники наблюдений

Высокоэффективная, но довольно сложная методика анализа пользователей, основанная на длительном по времени наблюдении за их действиями при работе с автоматизированной системой. Все действия фиксируются в виде дневниковых записей (в бумажном или электронном виде), в конце эксперимента производится анализ полученной информации. При достаточном объеме данных можно (и нужно) провести статистические исследования и получить количественные значения качественных показателей (например, через количество обращений к определенной операции оценить ее доступность через пользовательский интерфейс). Этот метод также подходит для анализа социальных связей и коммуникационных шаблонов внутри и между группами пользователей.

Сложности метода связаны, в основном, с нежеланием пользователей сотрудничать. Если дневник ведет наблюдатель-представитель разработчика, то он должен «слиться с фоном», поскольку мало кто из наблюдаемых любит, когда у него «стоят над душой». Если же дневник поручено вести самому пользователю, то часть информации он, скорее всего, «возьмет с потолка» (попробуйте проанализировать эту ситуацию самостоятельно).

Прототипирование

Прототипирование (создание прототипа) выполняется на основании результатов ранее произведенных исследований. Это позволяет всем заинтересованным сторонам оценить глубину проработки проекта, сравнить альтернативные варианты с учетом мнения заинтересованных сторон и выбрать то решение, которое пойдет в дальнейшую разработку.

Юзабилити-тестирование

Тестирование системы целевыми пользователями, которое может применяться на разных этапах ее создания. На ранних стадиях этот метод может быть применен в ходе анализа конкурирующих продуктов. При этом на пользователей возлагают задачи субъективной оценки и сопоставления предложений. Юзабилити-тестирование прототипов (в т.ч. и бумажных) позволяет оперативно и с меньшими затратами корректировать дизайн пользовательского интерфейса. При создании приложений, ориентированных на пользователей, юзабилити-тестирование входит в состав основного набора тестов, которые должны быть выполнены до передачи программного продукта в эксплуатацию.

Средства

Существует довольно большое количество инструментов, используемых в различных методах анализа пользователей. Среди них как офф-лайновые решения (начиная с обычной бумаги), так и он-лайновые сервисы. В табл. 1 приведены некоторые примеры программ, используемых при разработке пользовательских интерфейсов.

UI tool

UXSort

UXSort — Windows-приложение, позволяющее выполнять исследования, связанные с определением структуры методом сортировки карточек. Поддерживает до 1000 карточек, глубина сортировки — до 2-х уровней. Позволяет импортировать карточки из MS Excel или MS Word.

UI tool

Pencil Project (Evolus Pencil)

Свободная программа для создания прототипов, доступая для всех платформ. Легка в установке и использовании. Имеет большое количество подключаемых наборов шаблонов. Поддерживает экспорт в форматы .html, .svg, .pdf, .odt, .png.

UI tool

GUI Machine

GUI Machine — кроссплатформенный инструмент прототипирования интерфейсов десктопных и веб-приложений, позволяющий быстро и просто создавать высококачественные прототипы и просматривать их в интерактивном режиме. Содержит большое количество нативных и платформо-независимых компонентов.

UI tool

Moqups

Веб-приложение для создания прототипа сайта или мобильного приложения. Является удобным онлайн-редактором, для начала работы с которым даже не требуется регистрация. Доступны как бесплатная версия, так и коммерческая, с расширенными возможностями.

# **Прототипирование и концептуальное проектирование**

**Создание прототипа — следующий шаг в разработке пользовательского интерфейса. Прототипирование позволяет создавать макеты интерфейсов разной степени достоверности: от набросков «на скорую руку» и бумажных прототипов до интерактивных макетов с использование специальных программ.**

Глобальные прототипы моделируют систему целиком. Их использование позволяет выявлять проблемы, связанные с полнотой и непротиворечивостью пользовательского интерфейса.

Локальные прототипы моделируют только небольшую часть системы. Они могут быть использованы для устранения разногласий во мнениях через сопоставление различных вариантов дизайна: достаточно сделать несколько альтернатив и оценить их.

Два вида локальных прототипов, горизонтальные и вертикальные, направлены на полноту функциональности и диапазон возможностей в прототипе. Горизонтальные прототипы имеют небольшую функциональную глубину, но широки в возможностях. Вертикальные прототипы функционально глубоки, но ограничены в возможностях. Для лучшего понимания приведем две диаграммы, иллюстрирующме отличия горизонтального и вертикального прототипов для некоторой структуры приложения.

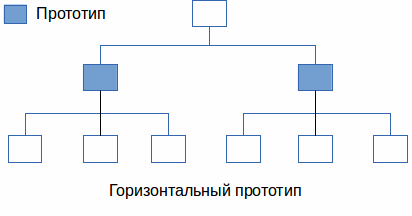
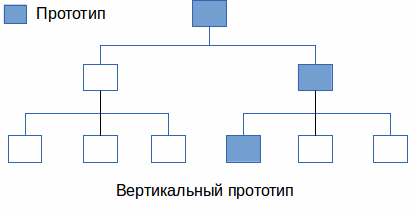
  


Рис. 1. Прототипы: горизонтальные и вертикальные

### Достоверность

Все прототипы можно разделить на прототипы низкой достоверности и прототипы высокой достоверности.

Первые, как правило, мало похожи на окончательный продукт. Они делаются не из того же самого материала, что и окончательное устройство и не имеют всей его функциональности. Прототип низкой достоверности может симулировать некоторую интерактивность, но не отражает всех тонкостей взаимодействия.

Вторые, напротив, выглядят более похожими на законченное устройство. Окончательный, утвержденный дизайн — пример прототипа высокой достоверности. Он может иметь некоторые из функций завершенного продукта и позволяет протестировать больше тонкостей взаимодействия. При этом, он требует больше времени на разработку и создание. Зачастую, после создания действующего прототипа высокой достоверности команда разработчиков или менеджеров неохотно отказывается от него и пытается развить из прообраза законченное устройство. В этом есть минус — такой подход может вести к ограниченному прогрессу в дизайне.

Каждая из указанных разновидностей имеет сильные и слабые стороны (Табл. 1).

Табл. 1. Преимущества и недостатки прототипов разной степени достоверности

|  |  |  |
| --- | --- | --- |
| **Достоверность прототипа** | **Преимущества** | **Недостатки** |
| Низкая | * Меньшая стоимость разработки * Возможность оценки множества вариантов дизайнов * Представляет полезную информацию для разработчиков и дизайнеров * Решает проблемы создания макетов экрана * Моет использоваться для определения потребностей рынка * Доказывает или опровергает идеи и концепции | * Ограниченный контроль ошибок * Плохая детализация спецификаций для дальнейшей разработки * Процессом управляет «посредник» (см. [facilitator-driven](https://www.google.ru/search?q=facilitator-driven+methodology&ie=utf-8&oe=utf-8&aq=t&rls=org.mozilla:en-US:official&client=firefox-a&channel=fflb&gfe_rd=cr&ei=SO9uVKPBBe21wAPV1YDwAQ)) * Ограниченная полезность после утверждения требований * Ограниченная пригодность для юзабилити-тестирования * Ограничения, связанные с навигацией и потоками активности |
| Высокая | * (Почти) полная функциональность * Интерактивность * Процесс разработки, управляемый пользователем * Четкая навигационная схема * Является инструментом для исследования и тестирования * Наглядно показывает конечный продукт * Служит «живой» спецификацией * Маркетинговый и торговый инструмент | * Высокая стоимость разработки * БОльшие затраты времени на создание * Не эффективен для проверки идей и концепций * Не эффективен для формирования окончательных требований |

**Полезный совет:** до начала создания прототипа высокой достоверности убедитесь, что ваш дизайн хорош, создав несколько прототипов низкой достоверности. Они могут принимать разные формы и каждая из них позволяет подтвердить или проверить различные аспекты дизайна.

## **Прототипирование**

Под прототипированием следует понимать ничто иное, как процесс создания прототипа. Хикс и Хартсон (Developing User Interfaces: Ensuring Usability through Product and Process, Wiley, 1993) описывают прототипирование по аналогии с артиллерийской стрельбой: артиллеристы наводят орудие в общем направлении на мишень; следует выстрел; корректировщики оценивают попадание (или промах) и радируют о необходимости корректировки прицела; после регулировки делается следующий выстрел. Так же и прототипирование: чтобы протестировать дизайнерскую идею (попадание или промах) нужно создать и оценить прототип.

### Виды прототипирования

Традиционный подход к разработке макета пользовательского интерфейса основан на переходе от прототипа низкой достоверности к прототипу высокой достоверности (рис. 2). На практике, эта простая и логичная схема выливается в более совершенную технологию эволюционного прототипирования.

[http://academy.kgtk.ru/it3/user-interfaces/img/traditional-prototyping-model_thumbnail.jpeg](http://academy.kgtk.ru/it3/user-interfaces/img/traditional-prototyping-model.gif)

Рис. 2. Традиционная модель разработки прототипа

**Эволюционное прототипирование** предполагает последовательное увеличение достоверности исходного образца, пока, в конце концов, он не становится законченной системой (рис. 3). Эволюционное прототипирование — широко распространенный подход к разработке интерфейсов, но он таит некоторую опасность: если изначально создается прототип высокой достоверности, его сложно будет расширять для проверки новых идей. Несмотря на это, эволюционное прототипирование может быть полезным для выявления все больших и больших тонкостей в аспектах дизайна и его совершенствования.

[](http://academy.kgtk.ru/it3/user-interfaces/img/evolutionary-prototyping-model.gif)

Рис. 3. Эволюционное прототипирование

**Быстрое прототипирование** подразумевает, что серии прототипов создаются, а затем, после их оценки и принятия решения о неадекватности модели, отбрасываются. Обычно разрабатываются прототипы все более и более высокой достоверности. Быстрое прототипирование может быть сложным для проектирования командой разработчиков или для приемки менеджерами, потому что выглядит так, как если бы время затраченное на разработку прототипа было потрачено впустую.

**Инкрементное прототипирование** основано на сборке окончательного продукта из нескольких прототипов. Все части (отдельные прототипы) могут разрабатываться параллельно, тем самым сокращая общее время на разработку (рис. 4).

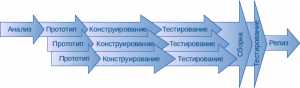
[](http://academy.kgtk.ru/it3/user-interfaces/img/incremental-prototyping-model.gif)

Рис. 4. Инкрементная модель

**Экстремальное прототипирование** (рис. 5) используется при создании веб-приложений. Весь процесс разбивается на три фазы. В первой фазе создается прототип низкой достоверности, состоящий из статических веб-страниц. Во второй фазе создается работоспособный код веб-приложения, а статические веб-страницы переписываются с учетом используемого фреймворка и функциональности, создается полностью работоспособный на уровне модели пользовательский интерфейс. В третьей фазе выполняется интеграция веб-интерфейса со всеми сервисами и ресурсами.

[](http://academy.kgtk.ru/it3/user-interfaces/img/extreme-prototyping-model.gif)

Рис. 5. Экстремальное прототипирование

## **Разработка прототипа: от теории к практике**

### Бумажное прототипирование

Вы имеете всего лишь несколько дней, оставшихся до представления ваших идей относительно приложения. Нет времени на создание полной модели (которая, на самом деле, есть другой прототип). Что вы можете сделать? Вы можете создать **бумажный прототип**.

Д. МакКракен и Р. Вольф в своей книге User-Centered Website Development: A Human-Computer Interaction Approach описывают очень гибкую и эффективную процедуру прототипирования веб-сайтов с помощью бумажного прототипа. Этот метод позволяет смоделировать не только визуальные аспекты веб-сайта, но и некоторые особенности взаимодействия с ним. Этот же метод может быть использован для разработки дизайнов приложений и небольших устройств.

Все, что вам потребуется для создания бумажного прототипа обычно есть под рукой:

* картон, линованная и нелинованная бумага, разноцветная бумага для заметок (sticky-notes), листы прозрачной ацетатной пленки;
* цветные ручки, фломастеры и карандаши;
* канцелярский клей, клейкая лента, клей многократного применения (подобный тому, который наносится на клейкие заметк);
* ножницы, канцелярский нож, хорошая линейка, циркуль.

Вы можете не иметь всех перечисленных принадлежностей, но стоит собрать большинство из них, прежде чем начнете.

В своей книге о проектировании веб-сайтов, Д. МакКракен и Р. Вольф предлагают в качестве основы для прототипа всех страниц сайта использовать распечатанный на плотной бумаге скриншот окна браузера. Они же предлагают делать все, что изменяется или исчезает на сайте (панели меню, полосы прокрутки, выпадающие меню, текстовые поля, текстовые блоки и т.п.) из бумаги. В Интернете есть множество готовых шаблонов указанных элементов, которые нужно просто скачать, распечатать, вырезать и использовать (например, [шаблоны ОС Андроид](http://developer.android.com/design/downloads/index.html)). Затем вырезанные из бумаги элементы страницы нужно разместить на шаблоне. По-разному компануя элементы макета или заменяя их на другие, вы, в конце концов, найдете подходящий вариант дизайна. Чтобы зафиксировать его, вы можете отсканировать полученные макеты или сфотографировать их.

Этот прототип позволяет моделировать интерактивность путем перемещения, удаления и размещения различных элементов на шаблоне.

**На заметку:** МакКракен и Вольф предлагают установку крайнего срока для изготовления прототипа, потому что многие участники команды разработчиков постоянно будут предлагать какие-нибудь усовершенствования в ходе работы над моделью. Это творческое усердие дизайнеров может длиться вечно, пока у них будут идеи, которые могут быть проверены.

### Раскадровки

Раскадровки часто используются в фильмах. Видели ли вы дополнительные возможности на DVD-диске фильма «Властелин колец»? Питер Джексон сделал раскадровки всего фильма, чтобы проиллюстрировать его исполнительным продюсерам и разработать сюжет. Вы можете спросить: что общего между кинематографом и интерактивным взаимодействием? Ответ в том, что и то, и другое — дискретные процессы, растянутые во времени.

Раскадровка — это последовательность зарисовок, показывающих, как пользователь продвигается «сквозь» задачу, используя конкретное устройство. Это могут быть эскизы графического пользовательского интерфейса (GUI), скриншоты ([как сделать скриншот экрана](http://prostocomp.net/soft/kak-sdelat-skrinshot-ekrana-na-kompyutere.html) на различных устройствах) или наброски сцен пользовательского взаимодействия с программой или устройством. Раскадровки очень хороши для «оживления» сценариев взаимодействия.

Для того, чтобы создавать раскадровки интерфейса не обязательно быть художником. Здесь не нужна фотореалистичность, поскольку каждая сцена раскадровки является, в первую очередь, визуальным средством общения. Концентрируйтесь на общении, а не на красоте рисунка. Это должно помочь вам думать о раскадровке, как о совокупности пиктограмм, каждая из которых обозначает объект или действие. Ни что не мешает вам нарисовать пользователя в виде фигурки из черточек, а компьютер — в виде прямоугольника.

Если вы используете для раскадровки какой-нибудь графический редактор, типа «GIMP» или «Photoshop», то очень важно не поддаться соблазну сделать ваш макет идеальным. Помните, дело не в том, чтобы получить красивую картинку, но в том, чтобы приблизить изображение интерфейса настолько хорошо, чтобы его хватило для дальнейшей работы над прототипом.

### Инструменты для создания макетов и каркасных моделей

Для создания прототипов часто используют специализированное программное обеспечение. Лучшие приложения для макетирования и каркасного моделирования (wireframing) позволяют моделировать интерактивность и делать раскадровки. Существуют приложения, которые позволяют демонстрировать подготовленные макеты команде разработчиков или клиентам через общий доступ. В онлайн-журнале [SpeckyBoy](http://speckyboy.com/2011/02/23/10-completely-free-wireframing-and-mockup-tools/) вы можете найти целый список инструментов макетирования и каркасного моделирования. В принципе, для создания

### Больше прототипов

Россон и Кэррол (Mary Beth Rosson, John M. Carroll. Usability Engineering: Scenario-Based Development of Human-Computer Interaction) предлагают следующий список различных способов прототипирования:

Табл. 2. Несколько разновидностей прототипов

| **Тип** | **Описание** |
| --- | --- |
| Раскадровка | Наброски или снимки экрана, иллюстрирующие ключевые точки в описаниии использования |
| Картонный макет | Образец устройства с имитацией управления или экранных элементов. Выполняется не обязательно из картона. |
| «Волшебник Оз» | Рабочая станция, связанная с невидимым человеком-асситентом, который симулирует ввод, вывод и исполняемую функциональность, которая еще не доступна |
| Видео-прототип | Видеозапись людей, разыгрывающих одну или несколько предполагаемых задач |
| Компьютерная анимация | Переходы экранов, которые иллюстрируют последовательности входных и выходных событий |
| «Машина сценариев» | Интерактивная система, реализующая определенный поток событий сценария |
| Быстрый прототип | Интерактивная система, созданная с помощью специальных инструментов прототипирования (в частности, с помощью визуальных средств разработки) |
| Частично работающая система | Исполнимая версия системы с ограниченной функциональностью |

**Вопросы:** Какие из приведенных в табл. 2 прототипов низкой достоверности, а какие — высокой? Какой из аспектов дизайна может проверить каждый из них? Насколько труден каждый из них в изготовлении? Какова выгода в стоимости для каждого прототипа?

## **Концептуальный проект**

В промышленности концептуальный проект понимается как специфичный документ, отражающий интегрированные идеи, описывающие систему: ее функции, внешний вид и порядок взаимодействия с пользователем. Но дизайнеры часто вкладывают в понятие концептуального проекта несколько иной смысл, подразумевающий идею системы в целом.

Прис, Роджерс и Шарп в своей книге о пректировании взаимодействия (Yvonne Rogers, Helen Sharp, Jenny Preece. Interaction Design: Beyond Human — Computer Interaction) предполагают, что концепция может быть основана на трех перспективах:

* **Деятельность** — действия, которые пользователи выполняют чаще всего, в терминах четырех парадигм:
  + указание инструкций (как в большинстве программ: пользователь указывает системе, что она должна делать, задавая нужные команды через меню или прямым вводом);
  + общение с системой (диалоговый режим);
  + манипуляция и навигация по системе;
  + исследование и просмотр;
* **Объекты** — продукты интерфейса (отображаемые результаты) или объекты, используемые в интефейсе.
* **Метафоры** — аналоги объектов или процессов реального мира.

Хороший концептуальный проект базируется на этих трех перспективах. Это можно проиллюстрировать примером. Один из наиболее востребованных видов офисных приложений — редакторы таблиц. Первый табличный редактор, VisiCalc, спроектировал Дэн Бриклин. До появления табличных процессоров, бухгалтеры вели учет, используя бухгалтерские книги и выполняя вычисления вручную. Бриклин понял, что бухгалтеры имеют проблемы с существующими инструментами, что компьютеры могут сделать процесс более производительным и интерактивным. Концептуальный проект будущего программного продукта был таким:

1. табличный процессор должен напоминать раскрытую бухгалтерскую книгу,
2. должна быть возможность редактировать содержимое ячеек таблиц,
3. должна быть возможность выполнять вычисления для диапазонов ячеек.

Его концептуальный дизайн учитывал деятельность (вычисления над данными, представленными в табличном виде), объекты (общие записи, ячейки, колонки) и имел хорошую метафору.

Другой пример — интерфейс «Звезда» (см. [Xerox 8010 Information System](http://en.wikipedia.org/wiki/Xerox_Star)) для офисных приложений, который был предшественником интерфейса типа «Рабочий стол».

**Вопрос:** Какая деятельность, объекты, метафоры в вашем концептуальном проекте?

**Метафоры** очень полезны в проектировании. Они связывают новый продукт с его предшественником. Они делают продукт более легким в освоении и использовании. Они могут помочь проектировщику в создании более согласованного дизайна интерфейса и выборе проектных альтернатив. Однако, метафора — прочная концепция и это может быть опасно. Пользователи могут верить в то, что новая система должна работать идентично аналогичной системе, с которой они уже знакомы и будут озадачены, если это окажется не так. Также проектировщики могут слишком сильно придерживаться метафоры, что может быть причиной плохого дизайна. Пример, иллюстрирующий и то, и другое: первые текстовые редакторы. В их основе — метафора с печатной машинкой: курсор, в виде символа подчеркивания или серого квадрата, отождествлялся с кареткой пишущей машинки. Пользователи, имевшие некоторый опыт работы с печатной машинкой, могли довольно быстро освоить текстовый редактор. Однако, у них часто возникали сложности с впечатыванием символов в пустые области или поверх пробелов. Метафора с печатной машинкой подсказывала им, что нужно поместить курсор в нужное, визуально не занятое какими-либо символами, место, чтобы напечатать там символ. Это то, что сделала бы печатная машинка, и это называлось «перевести каретку». Пользователи не могли осознать, что символ пробела — это такой же символ, как и другие. Так дизайнеры слишком сильно придерживались аналогии. Сам курсор в виде знака подчеркивания («\_») еще больше усиливал непонимание. Позже курсоры стали вертикальными линиями и это сделало понятней различие между метафорой и приложением. В данном примере все могло быть еще хуже: дизайнеры могли бы перемещать страницу вслед за вводом текста. Понимание того, что страница должна оставаться неподвижной позволило упростить создание текстовых документов, но, заметьте, пишущая машинка функционирует иначе. Здесь есть две морали:

* Подыщите хорошую метафору, но не придерживайтесь ее слишком сильно, делайте четкое различие.
* Попробуйте найти направления, в которых дизайн может улучшить старые способы ведения дел.

## **Вместо вывода**

Прототипирование и концептуальный дизайн — хоть и поставлены в индустриальные рамки, но все же процессы творческие. И чтобы творческое начало в них не угасло, проектирвщикам интерфейсов стоит придерживаться некоторых простых рекомендаций (индивидуально и в команде):

* Всегда держать разум открытым, никакой первоначальной критики любой идеи, которая может возникнуть.
* Принимать участие: постоянно обсуждать все, что вы знаете о проекте и окружающем его мире.
* Периодически форсировать генерацию идей, делая наброски, сочиняя стихи и песни.
* Сохранять идеи, пусть даже сумасбродные. Возможно, они станут основой для новых идей.

### 222222222

Статические методы тестирования.

Статические методы используются при проведении инспекций и рассмотрении спецификаций компонентов без их выполнения.Техника статического анализа заключается в методическом просмотре (или обзоре) и анализе структуры программ, а также в доказательстве их правильности. Статический анализ направлен на анализ документов, разработанных на всех этапах жизненного цикла и заключается в инспекции исходного кода и сквозного контроля программы.

*Инспектиция ПО*- это статическая проверка соответствия программы заданным спецификациями, проводится путем анализа различных представлений результатов проектирования (документации, требований, спецификаций, схем или исходного кода программ) на процессах жизненного цикла. Просмотры и инспекции результатов проектирования и соответствия их требованиям заказчика обеспечивают более высокое качество создаваемых программных систем.

При инспекции программ рассматриваются документы рабочего проектирования на этапах жизненного цикла совместно с независимыми экспертами и участниками разработки программных систем. На начальном этапе проектирования инспекция предполагает проверку полноты, целостности, однозначности, непротиворечивости и совместимости документов с исходными требованиями к программной системе. На этапе реализации системы под *инспекцией*понимается анализ текстов программ на соблюдение требований стандартов и принятых руководящих документов технологии программирования.

Эффективность такой проверки заключается в том, что привлекаемые эксперты пытаются взглянуть на проблему "со стороны" и подвергают ее всестороннему критическому анализу.

Эти приемы позволяют на более ранних этапах проектирования обнаружить ошибки или дефекты путем многократного просмотра исходных кодов. Символьное тестирование применяется для проверки отдельных участков программы на входных символьных значениях.

Кроме того, разрабатывается множество новых способов автоматизации символьного выполнения программ. Например, автоматизированное средство статического контроля для языков ориентированной разработки, инструменты автоматизации *доказательства корректности* и автоматизированный аппарат сетей Петри.

**Динамические методы тестирования**

*Динамические методы тестирования*используются в процессе выполнения программ. Они базируются на графе, связывающем причины ошибок с ожидаемыми реакциями на эти ошибки. В процессе тестирования накапливается информация об ошибках, которая используется при оценке надежности и качества программной системы.

Динамическое тестирование ориентировано на проверку корректности программных систем на множестве тестов, прогоняемых по программных системах в целях проверки и сбора данных на этапах жизненного цикла и проведения измерения отдельных элементов тестирования для оценки характеристик качества, указанных в требованиях, посредством выполнения системы на ЭВМ. Тестирование основывается на систематических, статистических, вероятностных и имитационных методах.

**Характеристика.** Систематические методы тестирования делятся на методы, в которых программы рассматриваются как "черный ящик" (используется информация о решаемой задаче), и методы, в которых программа рассматривается как "белый ящик" (используется структура программы). Этот вид называют тестированием с управлением по данным или управлением по входу-выходу. Цель – выяснение обстоятельств, при которых поведение программы не соответствует ее спецификации. При этом количество обнаруженных ошибок в программе является критерием качества входного тестирования.

///

Цель динамического тестирования программ по принципу "черного ящика" - выявление одним тестом максимального числа ошибок с использованием небольшого подмножества возможных входных данных.

*Методы "черного ящика"*обеспечивают:

* эквивалентное разбиение;
* анализ граничных значений;
* применение функциональных диаграмм, которые в соединении с реверсивным анализом дают достаточно полную информацию о функционировании тестируемой программы.

Эквивалентное разбиение состоит в разбиении входной области данных программы на конечное число классов эквивалентности так, чтобы каждый тест, являющийся представителем некоторого класса, был эквивалентен любому другому тесту этого класса.

Классы эквивалентности выделяются путем перебора входных условий и разбиения их на две или более групп. При этом различают два типа классов эквивалентности: правильные, задающие входные данные для программы, и неправильные, основанные на задании ошибочных входных значений.Разработка тестов методом эквивалентного разбиения осуществляется в два этапа: выделение классов эквивалентности и построение тестов. При построении тестов, основанных на выборе входных данных, проводится символическое выполнение программы.

Итак, методы тестирования по принципу "черного ящика" используются для тестирования функций, реализованных в программе, путем проверки несоответствия между реальным поведением функций и ожидаемым поведением с учетом спецификаций требований. Во время подготовки к этому тестированию строятся таблицы условий, причинно-следственные графы и области разбивки. Кроме того, подготавливаются тестовые наборы, учитывающие параметры и условия среды, которые влияют на поведение функций. Для каждого условия определяется множество значений и ограничений предикатов, которые тестируются.

*Метод "белого ящика"*позволяет исследовать внутреннюю структуру программы, причем обнаружение всех ошибок в программе является критерием исчерпывающего тестирования маршрутов потоков (графа) передач управления, среди которых рассматриваются:

* (а) критерий покрытия операторов - набор тестов в совокупности должен обеспечить прохождение каждого оператора не менееодного раза;
* (б) критерий тестирования ветвей (известный как покрытие решений или покрытие переходов) - набор тестов в совокупности должен обеспечить прохождение каждой ветви и выхода, по крайней мере, один раз.

Критерий (б) соответствует простому структурному тесту и наиболее распространен на практике. Для удовлетворения этого критерия необходимо построить систему путей, содержащую все ветви программы. Нахождение такого оптимального покрытия в некоторых случаях осуществляется просто, а в других является более сложной задачей.

Тестирование по принципу "белого ящика" ориентировано на проверку прохождения всех путей программ посредством применения путевого и имитационного тестирования.

*Путевое тестирование*применяется на уровне модулей и графовой *модели программы* путем выбора тестовых ситуаций, подготовки данных и включает тестирование следующих элементов:

* операторов, которые должны быть выполнены хотя бы один раз, без учета ошибок, которые могут остаться в программе иззабольшого количества логических путей и необходимости прохождения подмножеств этих путей;
* путей по заданному графу потоков управления для выявления разных маршрутов передачи управления с помощью путевых предикатов, для вычисления которого создается набор тестовых данных, гарантирующих прохождение всех путей. Однако все пути протестировать бывает невозможно, поэтому остаются не выявленные ошибки, которые могут проявиться в процессе эксплуатации;
* блоков, разделяющих программы на отдельные частиблоки, которые выполняются один раз или многократно при нахождении путей в программе, включающих совокупность блоков реализации одной функции либо нахождения входного множества данных, которое будет использоваться для выполнения указанного пути.

"Белый ящик" базируется на структуре программы, в случае "черного ящика", о структуре программы ничего неизвестно. Для выполнения тестирования с помощью этих "ящиков" известными считаются выполняемые функции, входы (входные данные) и выходы (выходные данные), а также логика обработки, представленные в документации.