

Рисунок 1 Три версии бумажного прототипа для домашнего термостата

## Стратегия для этой практики:

1. Выбрать конкретное направление реализации
2. Создать два быстрых цифровых прототипа
3. Оценить с помощью Эвристических правил.

## 1: Выберите раскадровку

В прошлом задании вы нарисовали две раскадровки. Проанализируйте их еще раз и выберите одну из двух раскадровок., как идею для прототипирования интерфейса. Определитесь с вашей точкой зрения.

## 2: Создайте два быстрых, интерактивных прототипа

Создайте два быстрых интерактивных прототипа. Этот прототип должен довольно точно показывает основные элементы пользовательского интерфейса. Это также называется Wire Frame или Wireframing.

Быстрый интерактивный прототип может быть сделан при помощи инструмента <https://ninjamock.com/>. Можно воспользоваться любым другим инструментом, позволяющим создавать прототипы высокого уровня. Не стоит использовать инструменты заставляющие писать код. Так же не стоит использовать Фотошоп, Иллюстратор и не нужно полировать работу, даже если вы очень хорошо владеете этими инструментами. Задача сейчас не в этом.

Создание этого прототипа не должно занять много времени.

Прототип должен быть интерактивным. Это значит, что по нему должна осуществляться навигация, и создаваться ощущение реальной работы. Это совсем не сложно. Такой инструмент как <https://ninjamock.com/> позволяет создавать линки между экранами на кнопках и элементах меню. Что позволит эмулировать такое поведение, как кнопки Home, Back, Next, Login. А так же реагировать на элементы меню.

Необходимо создать два прототипа пользовательского интерфейса, которые позволяют достичь одних и тех же целей, но принципиально разными путями. Не обязательно, полностью следовать раскадровке. В процессе прототипирования, вы можете немного переосмыслить поведение программы.

## Примеры

Ниже приведены две ссылки на прототипы, сделанные в инструменте Balsamiq Mockup, сделан экспорт в pdf, можно нажимать на некоторые кнопки, по ним будет осуществляться переход на другие страницы, таким образом эмулироваться работа приложения.

[1st prototype](https://coursera-uploads.s3.amazonaws.com/user-5d0ea56b1bdc4db63aa7130a/18/asst-2/18-4fd7ae7f6568b9.96995878.pdf)

[2nd prototype](https://coursera-uploads.s3.amazonaws.com/user-5d0ea56b1bdc4db63aa7130a/18/asst-2/18-4fd7bf94e602f7.45478747.pdf)

В <https://ninjamock.com/> можно осуществить такую же эмуляцию переходов.

## Артефакты

* Приложить к работе выбранную для прототипа раскадровку
* Два прототипа. В случае, если использовался <https://ninjamock.com/> то приложить ссылку на проект.
* Произвести эвристическую оценку, по 10 критериям для каждой формы интерфейса.

## Рекомендации

Желательно, придерживаться следующих рекомендаци:

* Прототипа должна быть направлена на удовлетворение пользовательских потребностей отображенных в раскадровке
* Идеи заложенные в прототип, должны удовлетворять потребности пользователя, способами максимально не похожими, на имеющиеся в общеизвестных сервисах. Должна быть оригинальная идея
* Два прототипа, должны быть максимально не похожи друг на друга, с точки зрения подходов к удовлетворению потребностей.

## Эвристическая оценка

Далее будут даны 10 критериев эвристической оценки от Якоба Нильсона. Интерпретация этих критериев ложиться на вас, в качестве помощи, в приложении 1, приведены возможные варианты интерпретации.

Что нужно сделать:

После того как будут нарисованы все интерфейсы, для каждой формы пользовательского интерфейса произвести оценку по 10 критериям.

Если какой-то из критериев не соблюдается необходимо записать в чем именно, приведя подробное описание в чем, возможно для удобство приложить скриншот.

Пример оформления:

**Прототип 1. Форма Логина**

|  |  |  |
| --- | --- | --- |
| **Критерий** | **Описание несоответствия** | **Скриншот (если нужно)** |
| Эстетика и минимализм | Слишком много информации, поля ввода логина и пароля убраны в угол, не сразу понятно как начать пользоваться системой |  |
| Помощь и документация | После неправильного ввода логина/пароля, выдается ошибка «Не получилось войти в систему». Не совсем понятно, что не получилось, может быть система не работает? |  |

**Прототип 1. Форма Окончания заказа**

|  |  |  |
| --- | --- | --- |
| **Критерий** | **Описание несоответствия** | **Скриншот (если нужно)** |
| Единообразие и стандарты | Единственная форма, на которой используется кнопка для перехода назад, в других случаях используется ссылка. |  |
|  |  |  |

**Прототипы. Общие замечания**

|  |  |  |
| --- | --- | --- |
| **№** | **Критерий** | **Оценка** |
| Прототип 1 | Ясность | 0 - Прототип отсутствует  1 - Большинство элементов не имеют ясной цели. Сложно понять для чего нужны большинство элементов.  3 - Некоторые элементы не имеют ясной цели. Не понятно, для чего нужны некоторые элементы  5 – Все элементы имеют четкое и понятное назначение. Легко понять, как используется прототип. Прототип можно отдать разработчику для реализации приложения. |
| Прототип 1 | Законченность | 0 - Прототип отсутствует  1 - Нет законченного прототипа. В нем нет понятной функциональности.  3 - Прототип в целом завершен, но присутствуют функции, которые не реализованы или не до конца продуманы.  5 – Прототип полностью завершен. Все требующиеся функции реализованы |
| Прототип 2 | Ясность |  |
| Прототип 2 | Законченность |  |
|  | Степень различия прототипов |  |
|  |  |  |

## Критерии для эвристической оценки

##### Отображение статуса системы

Система всегда должна информировать пользователя о том, что происходит - давать обратную связь в реальном времени.

##### Соответствие между системой и реальным миром

Система должна говорить с пользователем на понятном ему языке, понятными словами и фразами. Следите за тем, чтобы информация подавалась в логическом порядке.

##### Свобода действий и контроль

При использовании функций системы пользователи часто совершают ошибки и нуждаются в «аварийном откате», чтобы избежать непоправимых изменений. Настройте функции отмены и повтора.

##### Единообразие и стандарты

Однажды поняв принцип работы интерфейса, пользователь должен быть уверен, что в будущем система будет работать по тем же алгоритмам. Всегда следуйте одним и тем же правилам внутри продукта.

##### Профилактика ошибок

Лучше всякого сообщения об ошибках будет тщательная работа по их предотвращению. Помните об этом в первую очередь.

##### Видимость, а не переходы

Делайте объекты, опции и действия видимыми. Пользователь не должен запоминать информацию, переходя от одного объекта к другому. Инструкции по использованию системы должны быть видимыми или легкодоступными.

##### Гибкость и эффективность использования

Функции, не видимые новичкам, зачастую могут ускорить процесс взаимодействия опытных пользователей с сайтом. Упрощайте функции, которыми ваши клиенты пользуются чаще всего.

##### Эстетика и минимализм

Тексты не должны содержать явно лишнюю или неактуальную информацию. Каждая относительно бесполезная информация конкурирует с полезной и делает ее менее заметной.

##### Помогите пользователям распознавать, диагностировать и исправлять ошибки

Сообщения об ошибках должны быть выражены простым языком (никакого кода), точно указывать на проблему и предлагать её решение.

##### Помощь и документация

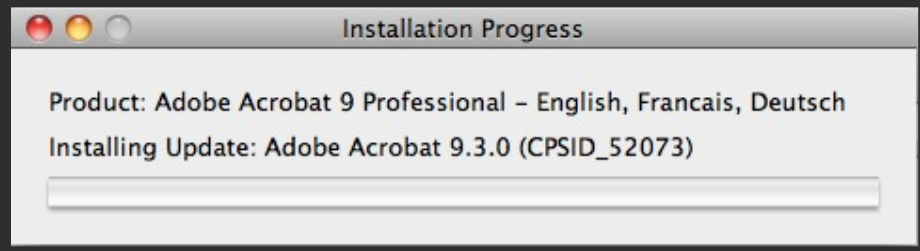
Систему можно использовать, не прибегая к помощи документации, однако не стоит ей пренебрегать. Любую подобную информацию должно быть легко найти, она должна быстро выводить пользователя на искомый вопрос и состоять из малого количества шагов.

## Приложение 1. Пояснение к эвристическим оценкам

### Отображение статуса системы

Одна из важнейших вещей в дизайне интерфейса - это обеспечить пользователю обратную связь, ясное представление о положении системы и его самого в общем процессе взаимодействия.

1. **Важный параметр этого представления - ориентация пользователя во времени.**



Если время отклика системы значительно меньше 1 секунды, то необходимость в промежуточной информации отпадает.

Когда же оно достигает одной-двух секунд, тогда становится важно подавать какие-нибудь сигналы о том, что система работает. Крутящийся значек вместо мышки, затемнение экрана… (какие еще???)

*(Байка про жалобы пользователе о долгой работе и о появлении статус бара, не скоративший реальное время, но убравший жалобы)*

Но если, время отклика превышает несколько секунд, то стоит сообщить пользователю о прошедшем времени и о предположительном времени окончания. Например, статус бар при установки программы. Или время скачивания файла. Иногда стоит выводить этапы процесса, как бы добавляя причастность пользователя к нему.

1. **Вторая важная категория. Сообщение о дисковом пространстве.**

Например, гугл почта постоянно извещает вас о том сколько у вас места используется.

1. **Третья категория. Показывать изменениях**

Скажем когда вы изменили документ, и собираетесь его закрыть не сохранив.

1. **Показывать. Действия.**

Самый шикарный пример из жизни – светофор. Красный стоп, зеленый поехал. При этом красный всегда сверху, зеленый снизу, что позволит ориентироваться даже людям с нарушением восприятия цвета. По сути тут используется избыточное (двойное) кодирование информации, но оно помогает в разных ситуациях.

1. **Показывать следующие шаги**

Очень важно подсказывать пользователю следующие шаги. Например, при покупке продукта в интернет магазине, вы решили уточнить, что то у продавца.

Хорошим сообщением после отправки запроса будет, как на амазоне: «Ваше сообщение отправлено продавцу, а копия была выслана на email-адрес, указанный в вашем профиле. Продавец пришлет вам свой ответ по электронной почте»

Вы сразу получили ответы на ряд вопросов: отправилось ли сообщение, получил ли кто-то ваше сообщение, как вам придет ответ, кто ответит.

1. **Результат**

После того как разобрались с ориентацией во времени, дисковым пространством, изменениями, действиями и следующими шагами, то стоит сообщить о результате.

Это может быть диалоговое окно. Экран в браузере однозначно говорящий о завершении. Возможно, использовать немного юмора. Веселая картинка, забавное сообщение. Поскольку завершение процесса это как завершение некоторой работы, несет облегчение.

### Говорить с пользователем на его языке. (Соответствие между системой и реальным миром)

Часто это означает создать попытку соответствия между интерфейсом и вещами в реальном мире. Например, MS Word – ведь такой интерфейс далеко не был первым. Сначала текстовые редакторы содержали в себе разметку, ключевые слова, заставляющие форматировать текст.

Еще это означает говорить на языке предметной области. А так же использовать образы общепринятые в онлайне или офлайне (они могут отличаться).

На интерфейсе Рисунок 1 можно увидеть целую вселенную, говорящую о том как будет напечатан документ.

Одним из преимуществ графического интерфейса в том, что он эксплуатирует метафоры из реального мира рабочий стол, корзина, файлы, папки. А если пользователь уже знаком с терминологией, то ему проще их ассоциировать с новой средой. Многие образы за последние 25 лет изжили себя, т.к. пользователям уже не надо так подробно проводить ассоциации. Наступила эпоха модернизма :) Появились новые метафоры.

Но при этом, не стоит забывать, что подавляющее большинство людей не программисты, поэтому не стоит их перегружать терминами. Например, попробуйте почитать какой-нибудь юридический документ или закон – это весьма трудоемкое занятие. При этом юристы ориентируются в них очень не плохо. Не стоит уподобляться им, а относится к пользователю с пониманием. Использовать простые термины и слова.

Важно, когда вы предоставляете какой-то выбор, что бы люди понимали о чем речь.

К примеру, редакторы текста при сохранении в формате UTF-8 могут спросить, сохранять его с BOM или без. Далеко не каждый в курсе, что такое метка порядка байт и для чего она нужна.

Конечно, решить это проблему невозможно, поскольку кодировки разные, но для пользователя, который не понимает в этом ничего, стоит указать наиболее распространенный вариант решения.

### Cвобода действий и удобство управления

Дайте пользователю возможность отмены ошибочных выборов, отмены и повтора действий, а не привязывайте его к рельсам жестких сценариев.

Вероятно, суть этого принципа - сделать так, чтобы достаточная свобода действий пользователя не мешала решению его задач.

Так, например, если интерфейс предполагается использовать **всего один раз**, то относительно **жесткий мастер** с подсказками будет вполне приемлемым решением. Однако, для более **продвинутых пользователей** лучше оставить возможность отклониться от этого жесткого сценария, чтобы совершить какие-нибудь **дополнительные** или специфические **действия** и настройки.

Например, на онлайн магазинах, сейчас практически везде введен пошаговый мастер, который однако позволяет вернуться к предыдущему пункту.

Нужно искать баланс между жестко зашитым мастером/путем и полной свободой действия, когда непонятны какие есть возможности.

### Последовательность и стандартизация

Поведение одинаковых элементов приложения должно быть одинаково и предсказуемо в разных его частях.

Назидательный пример аж 4 вариантов размещения набора кновок в диалоге Microsoft’s Visual Basic см Рисунок 2 VB пример 4 расположений групп кнопок.

Само собой если бы вариант был один, то было бы удобнее.

Так же необходимо быть последовательным в используемых словах и терминах. Не стоит использовать синонимы, они будут приводить к путанице.

Так же стоит пользоваться терминологией, которую используют пользователи. Но для этого необходимо будет провести исследование (полевые исследования, фокус группа, опрос).

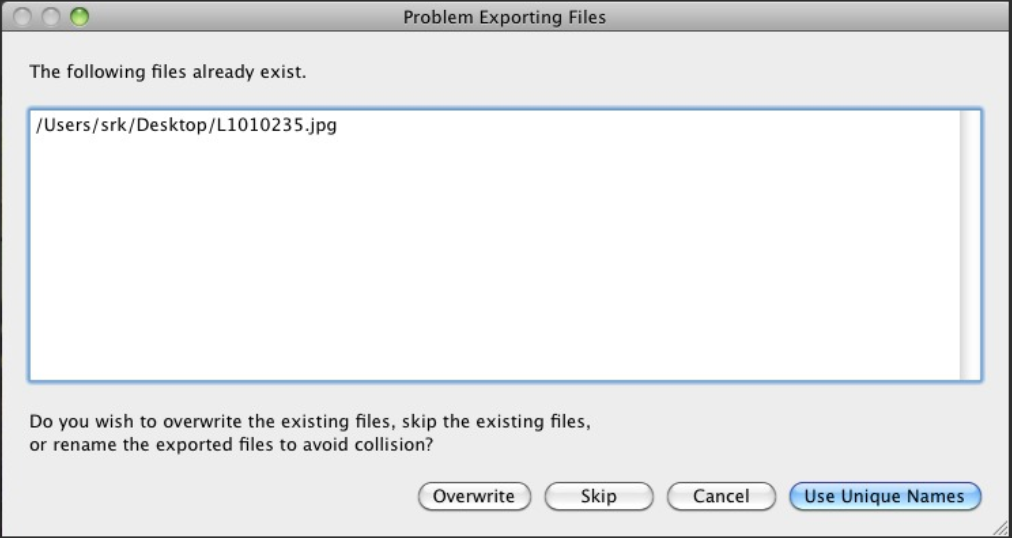
*Один из неудачных вариантов, когда в процессе диалога вам описывают ситуацию и задают вопрос «Вы хотите отменить?», Да/Нет. Обычно пользователь воспринимает Да как продолжить и это может ввести в заблуждение. Либо нужно изменить вопрос или если надо обратить особое внимание, то изменить Да/Нет, на Продолжить/Прекратить.*

### Предотвращение ошибок

Есть множество ошибок, которые легко совершить, и есть масса способов, которыми ваш интерфейс может уберечь людей от них.

1. Первый тип ошибки, от которой нужно подстраховать пользователя, - это потеря данных.

См. Рисунок 3 Диалог для предотвращения ошибки потери данных



Вот сообщение "Этот файл уже существует", и оно выполняет наше требование о конкретных названиях кнопок – отлично. Однако, я не знаю, как выглядит этот файл. Если бы мне предложили предварительный просмотр, я бы с гораздо меньшей вероятностью ошибся файлом. В дополнение к предотвращению потери данных вы также можете предотвратить распространение дубликатов.

Нужно находить компромисс между ограничениями и свободой действия.

**Уточнение**

Стоит отметить что многие интерфейсы могут иметь проблему относящуюся сразу к нескольким эвристикам, поэтому важнее определить проблему и придумать для нее решение. Если это удалось сделать то маркировка проблемы эвристикой не так важна.

### Узначание через воспоминание. Видимость, а не переходы

Создавайте интерфейсы, которые делают объекты, действия, опции и направления видимыми или легко восстановимыми.

### Гибкость и эффективность

Простой и тривиальный пример - предоставить специалистам возможность использовать клавиши быстрого доступа. Довольно удобно когда клавиши быстрого доступа подписаны в основном или всплывающем меню.

**Можно давать рекомендации**

Если в вашем приложении подразумевается выбор чего-либо из большого списка или даже поиск, то стоит предложить наиболее популярные варианты и только если они не подходят, уже осуществлять поиск через общий механизм.

Или например в онлайн магазинах можно порекомендовать похожий товар.

Однако проактивность должна быть уместной. Не стоит перегружать интерфейс.

Пример дистанционки устроенной поудобнее: Рисунок 4 Дистанционка

Гибкость и проактивность могут идти вразрез с принципами минимализма, поэтому во всем нужна мера.

### Эстетика и минимализм

Сразу пример: Ваш сайт предоставляет прогноз погоды, однако на не очень большом экране, саму погоду не видно, а видно только шапку сайта и другую вспомогательную информацию.

Понятно, что на «развороте» сразу видимом пользователю должна находится ключевая информация.

На сайте можно сделать кучу свистулек и рюшек, но нужно ли это пользователю. Например, на сайте поиска квартиры, совершенно ни к чему в личном кабинете возможность настройки виджетов. Да и возможно сам личный кабинет лишнее.

### Распознавание диагностирование и исправление ошибок

При обработке ошибки ввода пользователя главное точно и лаконично показать ему, что сделано не так.

К примеру, на форме ввода какое-то из полей заполнено не правильно. Раньше, в эпоху толстых клиентов, типовым сообщением было диалоговое окно с текстом. Конечно, информативно, но не столь удобно, особенно если пришлось заполнить два-три десятка полей, и ошибка допущена в трех из них. Сейчас для этого есть хороший паттерн, подсветка поля красным с пояснением ввиде хинта около поля.

В некоторых случаях есть возможность предотвратить неправильный ввод, задав маску ввода. Однако с масками стоит обращаться очень аккуратно, т.к. часто они вводят в заблуждение при вводе.

В случае, когда документ большой и ошибок может быть очень много, возможно стоит использовать лог с гиперссылками, перебрасывающих вас на нужную строку документа. И в случае исправления, убирающееся из лога.

Так же сообщение об ошибки должно не только описывать и разъяснять ее, а четко давать понять варианты ее исправления.

Не стоит заигрываться в сообщениях с системными/программистскими словечками. Например, если пользователь не ввел в какое-то поле значение, то не стоит сообщать ему текст. Поле не должно быть null.

### Помощь и документация

Проще всего думать о помощи как о проблеме не имеющую отношения к ПО. И в толстых клиентах это часто было так. К счастью, в случае с интернет приложениями, так поступать уже не принято.

К примеру, вы зашли на обучающий сайт и вам надо подписаться на рассылку по интересующей вас тематике. Довольно неудобно если подписка будет выглядет как выпадающий список с названием подписки и кнопкой подписаться. Более интересная стратегия разместить на странице все рассылки в виде блоков с краткой информацией, картинкой для более быстрой привязки и возможностью перейти на подробное описание. – назовем это **Стратегией предварительного просмотра**.

При использовании консольной утилиты, было бы не плохо при неправильном вводе параметров показать какие параметры вообще есть, или при их большом количестве указать команду получения подробной информации.

Когда вы показываете диалоговое окно, то стоит не только сообщать об ошибке, но и предлагать варианты решения, особенно если их не много.

Внедрение помощи прямо в приложение довольно мощный инструмент.

Например, на некоторых сайтах, при попытке отписаться от рассылки они предлагают уменьшить частоту получения писем, и возможно это сработает.