

Содержание

1 Перепись	2
1.1 Санта-клаус	2
1.1.1 Модель требований FRUPS+	2
1.1.2 Use-case диаграмма	3
1.1.3 Доменная модель	3
1.2 Disciplined Agile Delivery, +, -	4
1.2.1 Disciplined Agile Delivery	4
1.2.2 +	4
1.2.3 -	4
1.3 Эльфы и тестовое покрытие	4
2 Рубеж	5
2.1 Use-Case	5
2.2 Модель Ройса	5
2.3 Тестовое покрытие с использованием анализа эквивалентности	6
2.4 Ресурсные риски	6
2.5 Топ - 3 рисков по экспозиции	7
3 Термины	7
3.1 Модель требований FRUPS+	7
3.2 Доменная модель	8
3.3 Риски	8

1 Перепись

1.1 С помощью ИС «Санта-клаус» детям дарят подарки за хорошее поведение. Разработать модель требований, Use-Case модель и доменную модель.

1.1.1 Модель требований FRUPS+

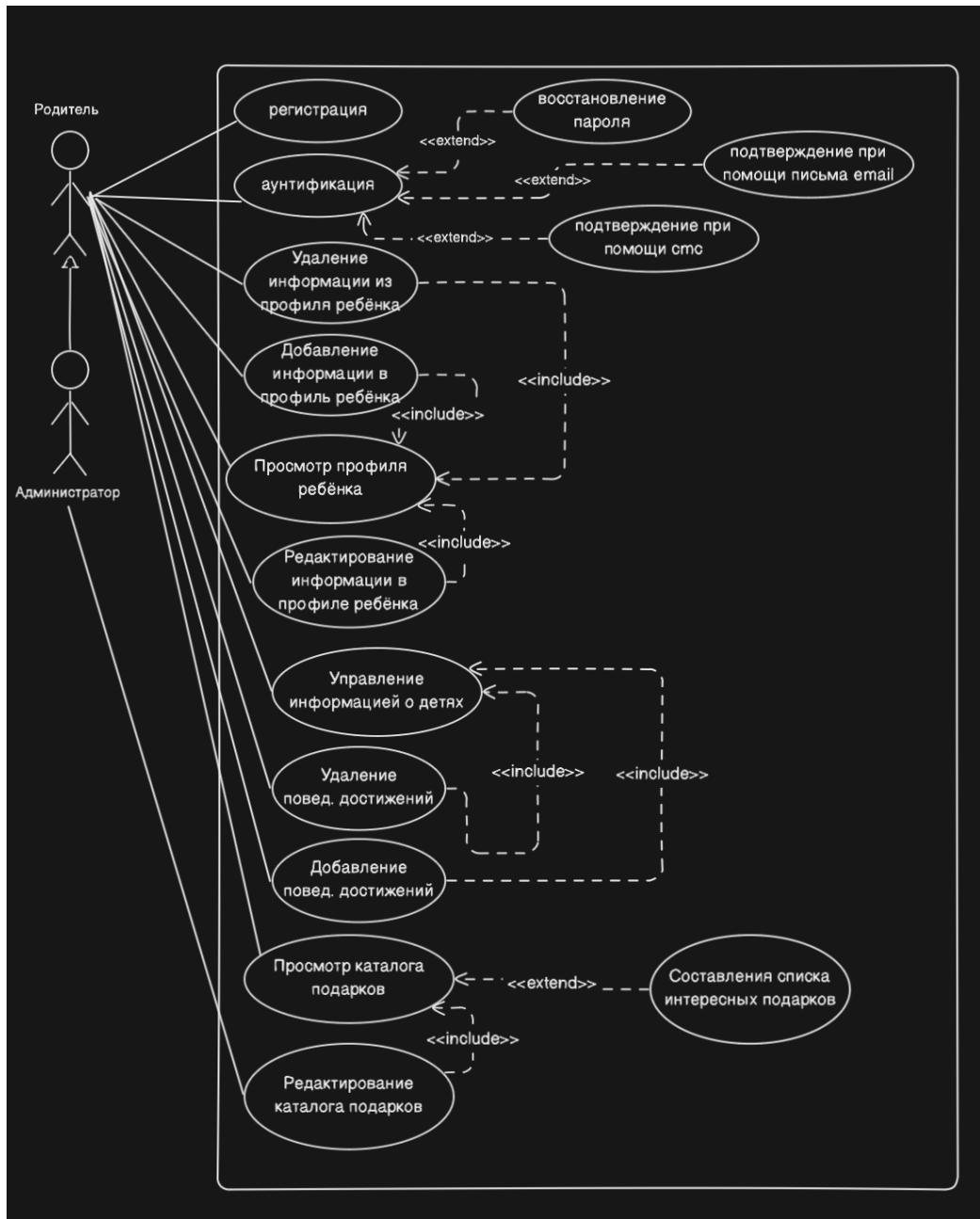
Функциональные требования

- SEC0 - Система должна обеспечивать регистрацию пользователей с помощью имени пользователя и пароля.
- SEC1 - Система должна поддерживать аутентификацию пользователей с помощью имени пользователя и пароля, также может использоваться подтверждения с помощью СМС или email письма.
- FR0 - Система должна поддерживать добавления, удаления, редактирование информации (не качества детей) о детях.
- FR1 - Система должна обеспечивать возможность отмечать, удалять поведенческие достижения их детей.
- FR2 - Система должна поддерживать просмотр каталога подарков.
- FR3 - Система должна поддерживать возможность составления списка интересных подарков для ребёнка.
- FR4 - Система должна предоставлять возможность восстановления забытого пароля.
- FR5 - Система должна поддерживать редактирование каталога подарков для админов.

Нефункциональные требования

- USA0 - Система должна обеспечивать адаптивный дизайн для различных устройств.
- RELI0 - Система должна быть доступна 99.9% времени, с автоматическим восстановлением после сбоев.
- PERF0 - Система должна обрабатывать тысячи запросов одновременно без существенных задержек.
- SUPP0 - Система должна легко масштабироваться для поддержки увеличения числа пользователей.

1.1.2 Use-case диаграмма



1.1.3 Доменная модель

Основные сущности:

- Пользователь (атрибуты: имя, роль, контактная информация, аутентификационные данные)
- Ребенок (атрибуты: имя, возраст, баллы поведения)
- Подарок (атрибуты: наименование, количество баллов для заказа, запасы)
- Заказ (атрибуты: дата, статус, связь с ребенком и выбранным подарком)

1.2 Модель разработки Disciplined Agile Delivery. «+» и «-» в сравнении с другими моделями

1.2.1 Disciplined Agile Delivery

Модель разработки разработанный Скоттом Амблером, который был вдохновлён системой деления на фазы (начало, построение и внедрение) и подходы как в RUP, но при этом основной цикл разработки построен на базе гибких методов, включая Scrum. Данная методология является относительно новой на рынке и только развивается. Так же DAD рассматривает процессы, которые выходят за его собственный процесс разработки (управление архитектурой и повторным использованием кода, управление персоналом, служба поддержки и текущих операций компании-разработчика, управление портфолио компетенций, непрерывное улучшение процессов разработки и вспомогательных процессов, и многое другое).

1.2.2 +

Одним из огромных плюсов (скорее даже это причина появления DAD) - на рынке требовалась методология, которая будет комфортна людям, которые, можно сказать, строят марсоход. Они не нуждаются в очень гибких методологиях, так как эти методологии не могут легко масштабироваться на большие команды разработчиков. Ещё плюсом является то, что можно комбинировать Agile и не Agile подходы в зависимости от требований проекта.

1.2.3 -

Минусы: сложность из-за большого количества инструментов и практик, отсюда ватикает следующий минус: высокий уровень компетенций команды для верного построения процессов.

1.3 Разработать тестовые сценарии (положительный и отрицательный) для следующего сценария

-Эльфы получают количество добрых дел у ребенка (K) по его имени и вводят его в систему;

-Если $K \geq 10$, то система назначает ему подарок

-Если $K < 10$, то система назначает ему уголек

-Эльф передает назначение на утверждение Санта-Клаусу

Оформить в виде таблицы тестовых случаев (Начальное состояние, ввод пользователя, вывод системы, конечное состояние)

Тест 1:

Начальное состояние: Готово и ожидает получение данных

Ввод пользователя: Иван, 10

Вывод системы: подарок

Конечное состояние: запрос на утверждение у Санта-Клауса

Тест 2:

Начальное состояние: Готово и ожидает получение данных

Ввод пользователя: Иван, 9

Вывод системы: уголёк

Конечное состояние: запрос на утверждение у Санта-Клауса

Тест 3:

Начальное состояние: Готово и ожидает получение данных

Ввод пользователя: Иван, -1

Вывод системы: неверный ввод данных

Конечное состояние: Готово и ожидает получение данных (Начальное состояние)

Тест 4:

Начальное состояние: Готово и ожидает получение данных

Ввод пользователя: Иван, "двадцать один"

Вывод системы: неверный ввод данных

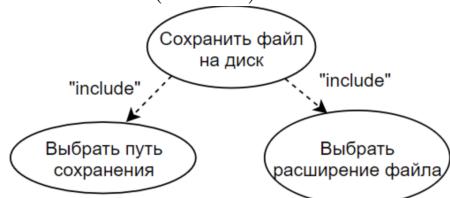
Конечное состояние: Готово и ожидает получение данных (Начальное состояние)

2 Вариант - 1

2.1 Use-Case

РАМКА БЕЗ АКТОРОВ!

Включение (Include) — обязательная, неотъемлемая связь между use-кейсами.



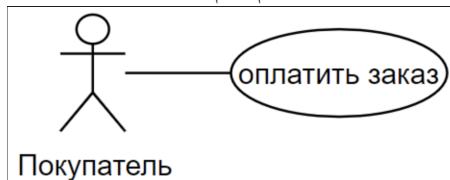
Расширение (Extend) — необязательное отношение. Если use-кейс не является обязательным, то актор может выбирать.



Отношение обобщения



Отношение ассоциации



2.2 Модель Ройса

Если кратко, то он предложил разбиение на следующие шаги:

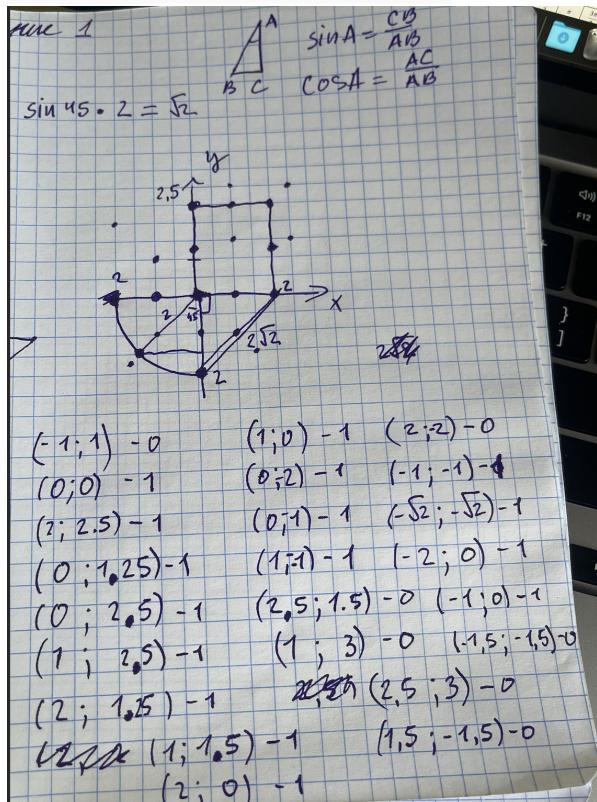
- Предварительный дизайн
- Документирование дизайна
- Just do it nike tw
- Тестирование
- Подключение пользователя

Подробнее

- Предварительный дизайн - занимаются только дизайнеры, цель: спроектировать, определить и создать модели обработки данных, даже если они будут требовать переделок. Разработать док - обзор будущей системы
- Документирование дизайна, которое включает: требования к системе, спецификация предварительного дизайна, спецификация дизайна интерфейсов, финальные спецификации дизайна системы, план тестирования, инструкция по использованию.

- Just do it nike tw - Предлагается запустить тестовую разработку параллельно основному процессу и взять это за каскадёра с сокращённым временем разработки, который позволит подтвердить основные характеристики ПО или найти ошибки.
- Тестирование - наиболее рискованная и дорогостоящая фаза, а также последний шанс для выбора альтернатив. Также при планировании тестирования исключается дизайнер и осмотр всех логических путей проводит другое лицо для инспекции кода. После исправления ошибок нужно провести checkout (тестирование)
- Подключение пользователя - это должно произойти на ранних этапах перед финальной поставкой продукта. Необходимо 3 точки: опыт, оценка и подтверждение пользователем - предварительный, критический и финальный просмотр.

2.3 Тестовое покрытие с использованием анализа эквивалентности



2.4 Ресурсные риски

Какие из перечисленных ниже рисков являются примерами ресурсных рисков? (перечислить номера вариантов ответа)

1. Риск поломки оборудования разработчиков.
2. Риск возникновения конфликта между менеджером исполнителя и заказчиком.
3. Риск отсутствия необходимых кадров при переносе производства в другой город.

4. Риск отсутствия спроса на произведенную продукцию.
 5. Риск запрета продажи автомобилей с ДВС для снижения негативного экологического воздействия на планету
- Ответ:
1. Риск поломки оборудования разработчиков.
 3. Риск отсутствия необходимых кадров при переносе производства в другой город.

2.5 Топ - 3 рисков по экспозиции

- Определите топ-3 рисков по экспозиции? Риски сортировать по уменьшению.

Номер риска	Название	Вероятность, %	Стоимость, млн руб.
1	Риск закрытия программы полётов на Марс на государственном уровне.	10%	20'000
2	Риск возникновения пылевой бури в атмосфере Марса в период посадки	25%	100'000
3	Риск использования в конструкции корабля неэффективных двигателей	20%	100'000
4	Риск нехватки запасов продовольствия на время полёта	15%	10'000
5	Риск выбора неудачного стартового окна для полёта	25%	20'000

Вероятность натсупления * потери = экспозиция

- $0.1 * 20000 = 2000$
- $0.25 * 100000 = 25000$
- $0.2 * 100000 = 20000$
- $0.15 * 10000 = 1500$
- $0.25 * 20000 = 5000$

Ответ: 235

3 Термины

3.1 Модель требований FRUPS+

Подробное описание того что должно быть реализовано системой, но при этом не должно описывать, как его нужно реализовать. Включает в себя функциональные требования, нефункциональные.

Функциональные определяют, что система должна делать: наборы функциональных требований (FR + номер), возможности ПО (CAP + номер), требования к безопасности (SEC + номер). Наборы - набор свойств продукты необходимый для выполнения конкретной деятельности (система должна обеспечивать ввод, модификацию и удаление данных о клиенте). Требования к безопасности - метод аутентификации, список ролей, шифрование, хранение данных в защищённых источниках (система должна обеспечивать двухфакторную аутентификацию пользователей с помощью имени пользователя и пароля и подтверждения с помощью СМС.)

Нефункциональные:

Usability - учёт особенностей пользователя (быстрота ответа в интервале), эстетические требования (цвет, расстояния между элементами), согласованность пользовательского интерфейса, согласованность пользовательского интерфейса, требования к справочной подсистеме, требования к пользовательской документации, требования к учебным материалам.

Reliability - частота и обработка заказов, способность системы восстанавливать продуктивное функционирование, предсказуемость поведения, точность, среднее время между отказами. Требования к надёжности, которые предназначены для способности ПО безотказно функционировать. В требованиях обычно указывается допустимое число отказов и сбоев за определённый промежуток времени. Способность системы восстанавливать продуктивное функционирование в течение заданного времени. Требованием является accuracy - точность, например, проведения вычислений. Коэффициент готовности системы — отношение времени исправной работы к сумме времён исправной работы и вынужденных простоев объекта, взятых за один и тот же календарный срок.

Performance - скорость решения задач, эффективность, готовность системы к решению задач, пропускная способность, время отклика, время восстановления, использование системных ресурсов. Требованием является скорость решения вычислительных задач. Также скорость важна в длительных инженерных расчетах, когда необходимо выполнить, например,

моделирование за разумное для человека время. Требования к эффективности фиксируют процент времени, которое тратится на выполнение полезных задач, по отношению к времени на выполнение общесистемных. Требованием к производительности является готовность (availability) быстро начать выполнение задачи. Какой объём данных или запросов система может обработать за единицу времени. Для большой реактивности придется пожертвовать пропускной способностью.

Supportability - способность системы к расширению и масштабированию и выполнению большего объема обработки данных. Адаптируемость под конкретные задачи, поддерживаемость. Требования к совместимости позволяют использовать различные операционные системы, версии продуктов, браузеров и пр. совместно с разработанным ПО. Отдельно выделяются системные требования и минимальные требования к установке системы, например, объём ОЗУ, количество и частота процессоров и пр.

3.2 Доменная модель

Доменная модель — это концептуальная модель предметной области, которая отображает ключевые сущности, их атрибуты и взаимосвязи между ними, а также основные правила бизнес-логики. Эта модель помогает разработчикам и всем участникам проекта лучше понять структуру и функционирование системы, на которой они работают. 33 слайд

3.3 Риски

- Прямые и непрямые - можем контролировать или управлять или нет. Прямыми рисками можно в явном виде управлять: воздействовать на них, уменьшать их вероятность, реагировать на них. Непрямые риски возникают по внешним причинам и не поддаются управлению, можно только принять на себя их последствия. Отсюда существуют различные методы работы с прямыми и непрямыми рисками.
- Ресурсные - организационные, финансовые, люди, время. связаны с недостатком у компании времени, людей, денег, оборудования. Ути риски имеют отношение к особенностям и специфике конкретной организации, разрабатывающей проект. Данные риски являются управляемыми, так как возможно изменение выделяемых ресурсов.
- Бизнес-риски - конкуренция, подрядчики, убыточность решения. появляются из-за взаимодействия с другими организациями и рынком в целом. Они определены конкуренцией, взаимодействием с подрядчиками или потенциальной убыточностью решения, т.е. отсутствием в дальнейшем прибыли от реализации и внедрения ПО. Управление бизнес- рисками сложно поддается управлению со стороны разработчиков.
- Технические риски - границы проекта, технологические, внешние зависимости. находятся в пределах компетенции разработчиков и поэтому являются ими управляемыми. Примеры технических рисков - отсутствие у разработчиков компетенции в применяемых технологиях.
- Форс-мажор. В отличие от политических рисков, повлиять на форс-мажор нельзя, и предугадать его тоже. К таким рискам относятся стихийные бедствия, изменение законодательства и т.п.
- Политические риски - связаны с изменением сфер влияния внутри компании-заказчика. Например, с появлением нового менеджера могут измениться условия сделки. Это возможно предвидеть, но не всегда. Противодействием может быть упреждающая реакция на возможные изменения в компании (например, установление контактов с вероятным новым менеджером).