Современные методы разработки ПО

Федоров Арсений
20П-3

Программная инженерия – систематический, дисциплинированный и измеримый подход к разработке программного обеспечения

Прикладная программная система (приложение) реализует набор взаимосвязанных функций некоторой предметной области. Например, программы расчета затрат, учета материалов, продаж, траекторий движения и т.д. Могут быть также общесистемные и специализированные

программные средства. Приведите примеры.

Способы изготовления программного обеспечения – это и есть **программная инженерия**. Понятие включает процессы жизненного цикла ПО, методы разработки и управления, методы оценки программных продуктов и процессов. И все это ради их постоянного совершенствования.

У любого программного обеспечения есть жизненный цикл — этапы, через которые оно проходит с начала создания до конца разработки и внедрения. Чаще всего это подготовка, проектирование, создание и поддержка. Этапы могут называться по-разному и дробиться на более мелкие стадии.

Рассмотрим эти этапы на примере жизненного цикла интернет-магазина.

- **Подготовка.** Иван решил запустить книжный интернет-магазин и начал анализировать, какие подобные сайты уже представлены в сети. Собрал информацию об их трафике, функциональности.
- **Проектирование.** Иван выбрал компанию-подрядчика и обсудил с её специалистами архитектуру и дизайн будущего интернет-магазина.
- **Создание.** Иван заключил с разработчиками договор. Они начали писать код, отрисовывать дизайн, составлять документацию.
- **Поддержка.** Иван подписал акт сдачи-приёмки, и подрядчик разместил интернет-магазин на «боевых» серверах. Пользователи начали его посещать и сообщать о замеченных ошибках в поддержку, а программисты — оперативно всё исправлять.

***Модель* разработки программного обеспечения** описывает, какие стадии жизненного цикла оно проходит и что происходит на каждой из них.

А *методология* включает в себя набор методов по управлению разработкой: это правила, техники и принципы, которые делают её более эффективной.

Рассмотрим

Основные модели разработки ПО

1 Каскадная модель, или «водопад»

разработка осуществляется поэтапно: каждая следующая стадия начинается после того, как заканчивается предыдущая. Если всё делать правильно, «водопад» будет наиболее быстрой и простой моделью

Преимущества «водопада»

- *Разработку просто контролировать.* Заказчик может управлять сроками и ценой.
- *Стоимость проекта определяется на начальном этапе.* Все шаги запланированы уже на этапе согласования договора, ПО пишется непрерывно.
- *Не нужно нанимать тестировщиков с серьёзной подготовкой.* Тестировщики смогут опираться на подробную техническую документацию.

Недостатки каскадной модели

- *Тестирование начинается на последних этапах разработки.* Если в требованиях к продукту была допущена ошибка, то исправить её будет стоить дорого. Обнаружат её, когда разработчик уже написал код, а технические писатели — документацию.
- *Заказчик видит готовый продукт в конце разработки и только тогда может дать обратную связь.* Велика вероятность, что результат его не устроит.
- *Пишут много технической документации, что задерживает работы.*

«Водопад» подходит для разработки проектов в *медицинской и космической отрасли,* ***где уже сформирована обширная база документов*** **(СНиПов и спецификаций),** на основе которых можно написать требования к новому ПО.

При работе с каскадной моделью **основная задача — написать подробные требования к разработке**. На **этапе тестирования не должно выясниться, что в них есть ошибка**, которая влияет на весь продукт.

2. V-образная модель (разработка через тестирование)

Это усовершенствованная каскадная модель, в которой заказчик с командой программистов одновременно составляют требования к системе и описывают, как будут тестировать её на каждом этапе.

Преимущества V-образной модели

Количество ошибок в архитектуре ПО сводится к минимуму.

Недостатки V-образной модели

Если при разработке архитектуры была допущена ошибка, то вернуться и исправить её будет стоить дорого, как и в «водопаде».

V-модель подходит *для проектов, в которых важна надёжность и цена ошибки очень высока. Например, при разработке подушек безопасности для автомобилей или систем наблюдения за пациентами в клиниках.*

3. Инкрементная модель

Это модель разработки по частям (increment в переводе с англ. — приращение)

Рассмотрим её на примере создания социальной сети.

1. Заказчик решил, что хочет запустить соцсеть, и написал **подробное техническое задание.** Программисты предложили **реализовать основные** функции — страницу с личной информацией и чат. А затем **протестировать на пользователях**.
2. Команда разработки **показывает продукт заказчику и выпускает его на рынок**. Если и заказчику, и пользователям социальная сеть нравится, работа над ней **продолжается, но уже по частям**.
3. Программисты **параллельно создают функциональность** для загрузки фотографий, обмена документами, прослушивания музыки и других действий, согласованных с заказчиком. Инкремент за инкрементом они совершенствуют продукт, приближаясь к описанному в техническом задании.

Недостатки инкрементной модели

•*Каждая команда программистов разрабатывает свою функциональность и может реализовать интерфейс продукта по-своему.* Чтобы этого не произошло, важно на этапе обсуждения техзадания (ТЗ) объяснить, каким он будет, **чтобы у всех участников проекта сложилось единое понимание**.

•*Разработчики будут оттягивать доработку основной функциональности.* Чтобы этого не случилось, **менеджер проекта должен контролировать, чем занимается каждая команда**.

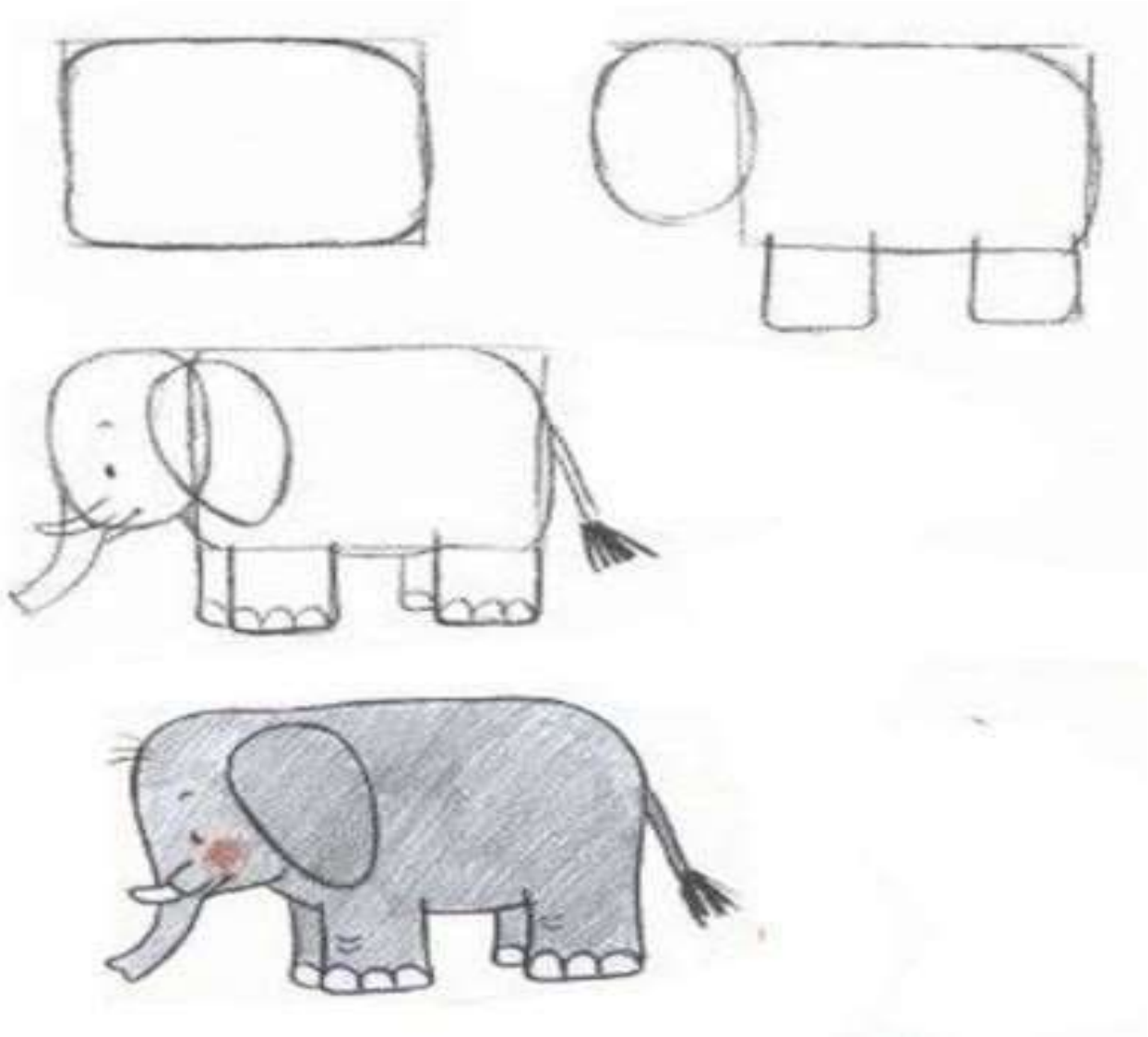
Инкрементная модель подходит для *проектов, в которых* ***точное техзадание прописано уже на старте****, а продукт должен быстро выйти на рынок.*

Преимущества инкрементной модели

- *Не нужно вкладывать много денег на начальном этапе.* Заказчик оплачивает создание основных функций, получает продукт, по итогам обратной связи решает, продолжать ли разработку.
- *Можно быстро получить ответ от пользователей и оперативно обновить техническое задание.* Снижается риск создать продукт, который никому не нужен.
- *Ошибка обходится дешевле.* Если при разработке архитектуры была ошибка, исправить её будет стоить не так дорого.

4. Итеративная модель

Модель, при которой **заказчик не обязан понимать, какой продукт хочет** получить в итоге, и может не прописывать сразу подробное техзадание.



Рассмотрим **на примере создания мессенджера**

1.Заказчик решил, что хочет создать мессенджер. Разработчики **сделали приложение**, в котором можно **добавить друга и запустить чат на двоих**.

2.Мессенджер «выкатили» в магазин приложений, пользователи начали его скачивать и активно использовать. Заказчик понял, что **продукт пользуется популярностью**, и решил его доработать.

3.Программисты **добавили в мессенджер возможность просмотра видео, загрузки фотографий, записи аудиосообщений**. Они постепенно улучшают функциональность приложения, адаптируют его к требованиям рынка.

Преимущества итеративной модели

- *Быстрый выпуск минимального продукта* даёт возможность оперативно получать обратную связь от заказчика и пользователей. А значит, фокусироваться на наиболее важных функциях ПО и улучшать их в соответствии с требованиями рынка и пожеланиями клиента.
- *Постоянное тестирование пользователями* позволяет быстро обнаруживать и устранять ошибки.

Недостатки итеративной модели

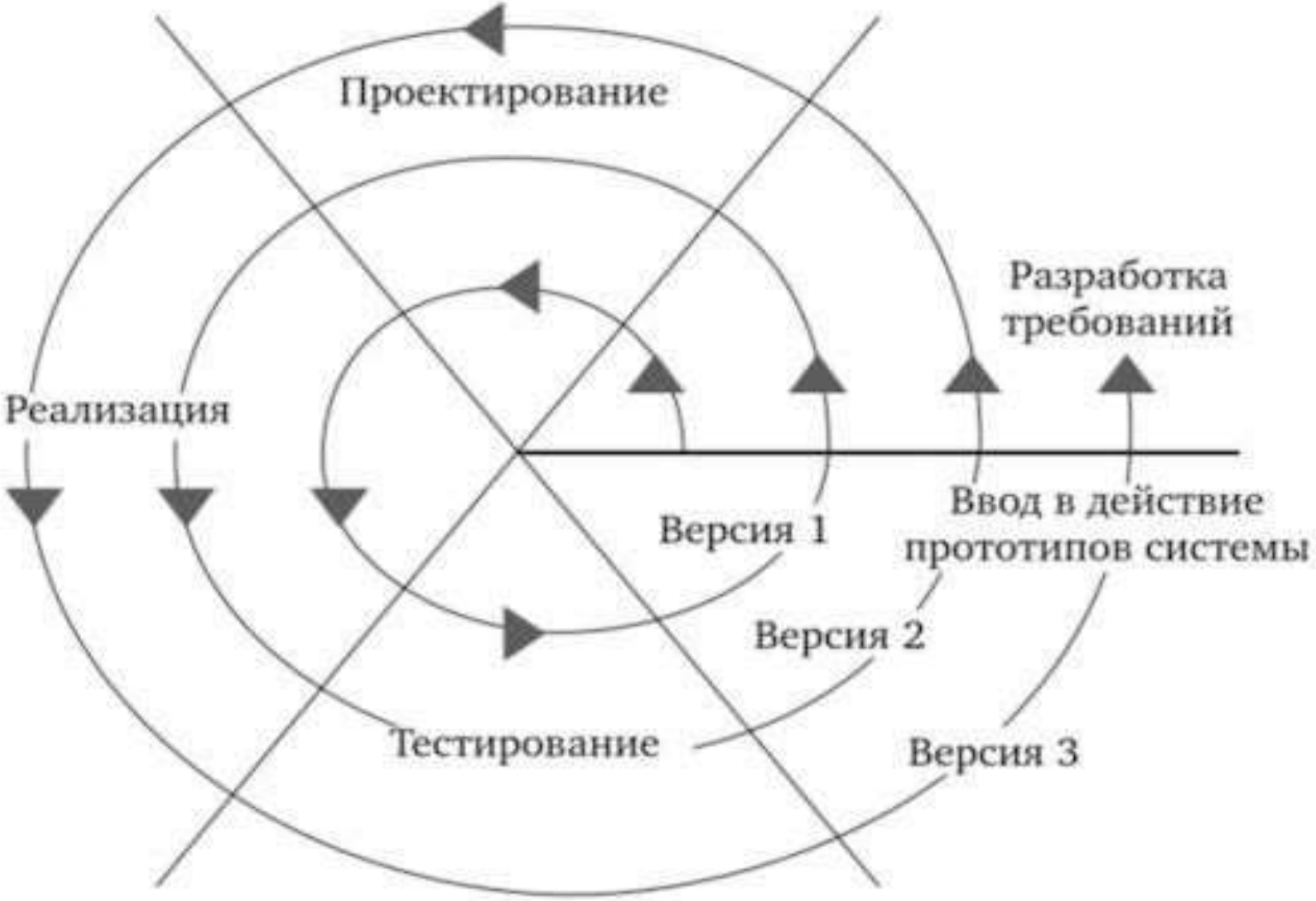
•*Использование на начальном этапе* ***баз данных или серверов*** — первые сложно масштабировать, а вторые не выдерживают нагрузку. Возможно, придётся переписывать большую часть приложения.

•*Отсутствие фиксированного бюджета и сроков.* Заказчик не знает, как выглядит конечная цель и когда закончится разработка.

Итеративная модель подходит для работы над *большими проектами с неопределёнными требованиями*, либо для задач с *инновационным подходом,* когда заказчик не уверен в результате. (Пример ПО для проведения чемпионатов WorldSkills)

5. Спиральная модель

заказчик и команда разработчиков серьёзно анализируют риски проекта и выполняют его итерациями. Последующая стадия основывается на предыдущей, а в конце каждого витка — цикла итераций — принимается решение , продолжать ли проект



Рассмотрим, как функционирует эта модель, **на примере разработки системы «Умный дом».**

1.Заказчик решил заказал программистам реализовать **управление чайником с телефона**. Они начали действовать по каскадной модели: провели анализ предложений на рынке, обсудили с заказчиком архитектуру системы, разработали, протестировали.

2.Заказчик оценил результат и риски: нужна ли пользователям следующая версия продукта – **управление телевизором**. Рассчитал сроки, бюджет и заказал разработку. Программисты по той же модели представили заказчику более сложный продукт на базе первого.

3.Заказчик решил создать функциональность для управления **холодильником с телефона**. **Но, анализируя риски, понял**, что в холодильник сложно встроить Wi-Fi-модуль, и производители не заинтересованы в сотрудничестве. Т.е. риски превышают потенциальную выгоду. Заказчик решил прекратить разработку и совершенствовать имеющуюся функциональность.

Спиральная модель похожа на инкрементную, но здесь гораздо больше времени уделяется **оценке рисков**. С каждым новым витком спирали процесс усложняется. Используется в *исследовательских проектах и там, где высоки риски.*

Преимущества спиральной модели

•*Большое внимание уделяется проработке рисков.*

Недостатки спиральной модели

•*Есть риск застрять на начальном этапе* — бесконечно совершенствовать первую версию продукта и не продвинуться к следующим.

•*Разработка длится долго и стоит дорого.*

На основе итеративной модели была создана

6. Agile — не модель и не методология, а скорее подход к разработке.

Что такое Agile?

Agile («эджайл») переводится с английского как «гибкий». Включает в себя практики, подходы и методологии, которые помогают создавать продукт более эффективно:

- **экстремальное программирование (**XP-все методы доводить до экстремальных: коллективное владение кодом, частые релизы, парное программирование, непрерывная интеграция, мах во всем…**)**
- **бережливую разработку программного обеспечения** (lean из производства – сокращение всех потерь)
- **фреймворк для управления проектами Scrum** (динамика, постоянное обсуждение, командный дух);
- **разработку, управляемую функциональностью**
- **разработку через тестирование**
- **методологию «чистой комнаты»** (сертифицируемый уровень надежности)
- **итеративно-инкрементный метод разработки**
- **методологию разработки Microsoft Solutions Framework**
- **метод разработки динамических систем** (постоянно меняется)
- **метод управления разработкой Kanban** (принцип «точно в срок»)

Не всё перечисленное в списке — методологии.

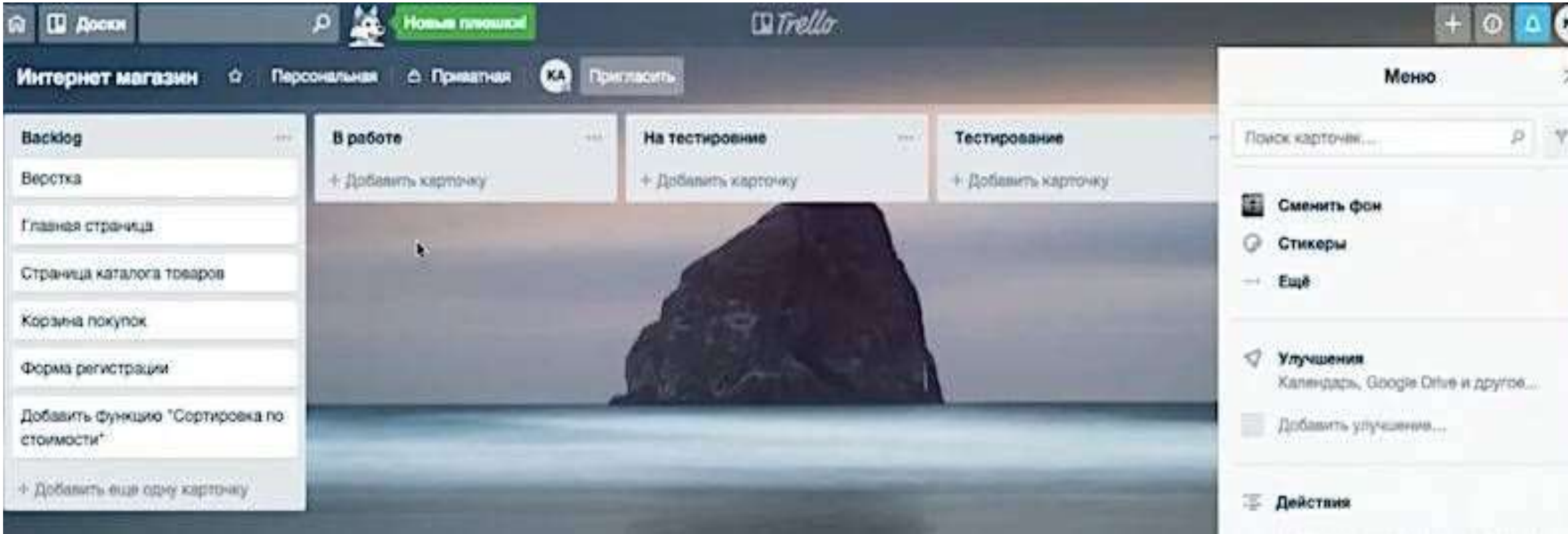
Например, Scrum чаще называют не методологией, а **фреймворком.** В чём разница? Фреймворк — это более сформированная методология со строгими правилами. В скраме все роли и процессы чётко прописаны. Помимо Scrum, часто используют Kanban.

Фреймворк (каркас, каркасный подход) -

программная платформа, определяющая структуру программной системы, программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

Kanban - одна из наиболее популярных методологий разработки ПО. Команда ведёт работу с помощью виртуальной доски, которая разбита на этапы проекта. Каждый участник видит, какие задачи находятся в работе, какие — застряли на одном из этапов, а какие уже дошли до его столбца и требуют внимания.

В отличие от скрама, в канбане можно взять срочные задачи в разработку сразу, не дожидаясь начала следующего спринта. **Канбан удобно использовать не только в такой работе.**



Различия между Agile и традиционным подходом к разработке

Характеристики проекта	Традиционные модели	Гибкие методологии
Подход	Прогнозирующий	Адаптивный
Критерии успеха	Следование плану	Ценность для бизнеса
Риски	Риски определены	Риски не определены
Контроль	Легко контролировать	Зависит от профессионального уровня специалистов
Заказчики	Низкая вовлеченность	Высокая вовлеченность
Документация	Детальная с начала проекта	Доработка по мере развития проекта
Требования	Известны заранее, стабильны	Не всегда известны заранее, легко изменяемы
Команда проекта	Включение новых специалистов на любом этапе	Опытные специалисты, стабильный состав
Рефакторинг (изменение кода)	Дорого	Недорого

бережливая, через тестирование, управляемая функциональностью, итеративно-инкрементная, …