# Вступление

Привет!

# Риски – это большая проблема

Риски:

1. Проваленный срок
2. Отмененный проект
3. Замена системы из-за непригодности в долгосрочном использовании
4. Огромное количество дефектов в готовом продукте
5. Созданный продукт не решает задач бизнеса
6. Бизнес изменился – пока делали полгода продукт, рынок поменялся
7. Большое количество задач было с точки зрения программирования, но не бизнеса
8. Люди уходят, не выдержав проекта

# Кривая затрат

Как меня учили в университете: делаем все последовательно.

Ошибка, исправленная на стадии формирования требований, будет стоить десятки рублей, та же самая ошибка, найденная на стадии тестирования, обойдется уже в десятки тысяч.

Мы должны согласиться с тем, что система будет изменяться со временем. Это неизбежно. Меняются требования пользователей, меняется бизнес.

Мы стремимся сделать кривую затрат пологой.

# Экстремальное программирование

Экстремальное программирование принимает то, что требования к системе меняются, за данность.

Почему «экстремальное»? Потому что все по максимуму.

# Игра в планирование

Планирование в XP сильно отличается от планирования в других методологиях. Обычные методологии пытаются заменить коммуникацию документацией.

Традиционный подход:

1. собрать все требования заказчика
2. планированием требует специальных навыков, поэтому этим должны заниматься профессиональные менеджеры
3. сначала нужно поговорить с заказчиком, затем выполнить планирование, а затем предоставить готовый план
4. поэтому планирование – весьма дорогостоящая процедура и должна выполняться как можно реже. Нужно хорошо проработать план и следить за его исполнением.

Если вы ведете машину и изначально задали ей направление движения, то, скорее всего вы окажитесь в кювете.

Планирование – механизм управления движением проекта.

Реальный подход:

1. В самом начале требования заказчика не могут быть полными
2. В процессе планирования должны принимать участия все имеющие отношение к проекту люди
3. Все заинтересованные стороны должны постоянно поддерживать коммуникацию
4. Планирование должно выполняться очень часто и требовать небольших затрат

В XP планирование – это часть процесса разработки, оно выполняется очень часто. Мы планируем выпуск версии раз в несколько месяцев, мы планируем следующую итерацию раз в несколько недель, мы планируем нашу работу на каждый день в ходе экспресс-совещания.

Процесс планирования:

1. Заказчик и разработчики собираются вместе
2. Заказчик рассказывает, что он хочет получить от создаваемой системы.
3. Разработчики записывают его пожелания на карточки, по ходу уточняя детали.
4. Заказчик сортирует созданные с его слов пожелания по категориям - очень важные (без них нет смысла работать следующие 2 недели), обязательные (без реализации этих пожеланий часть пользователей будет испытывать неудобства) и желательные.
5. Разработчики оценивают каждое пожелание, выставляя на каждой карточке время (1, 2, 4, 8), которое им придется потратить. Оценка происходит в идеальных днях - это такой день, когда все 8 часов разработчик занимается только этой задачей, ни на что другое не отвлекаясь. При этом заказчик уточняет, почему так много времени займет реализация того или иного пожелания.
6. Заказчик, исходя из расставленных на карточках сроков (цены пожелания), выбирает задачи на следующие 2 недели.

*Аналогия*  
Вы приходите в магазин и говорите: "Беру все!". Продавец указывает вам на ценники. Вы меняете свою точку зрения и говорите: "Ясно, тогда для начала я беру вот это и вот это". Вам приходится исходить из своих ресурсов при выборе покупки.

Карточка – это не документ, это лишь повод продолжить подробное обсуждение требования в ходе программирования.

# Экспресс-совещания

Никто не может знать абсолютно всего. Парное программирование частично решает эту проблему, но чем занимаются другие пары? У кого именно надо получить интересующую вас информацию?

Экспресс-совещания – коротное ежедневное собрание всех членов команды с целью обмена информацией о текущем состоянии дел.

Цель – соединить тех, кто владеет информацией и тех, кто в ней нуждается.

Каждый должен ответить на вопросы:

* Что я сделал вчера и что узнал нового?
* Как далеко я продвинулся в решении своей проблемы и какие возникли препятствия?
* Что я буду делать сегодня?

Причины, из-за которых собрания могут не проводиться:

* не набрать необходимого количества участников
* люди, обладающие информацией, не хотят ею делиться
  + информация наделяет их высоким статусом
  + боятся показать, что знают не много
* люди, обладающие информацией, пытаются сообщить присутствующим слишком много (углубляются в детали)
* нет места для проведения

# Частые выпуски

Версии продукта должны поступать в эксплуатацию как можно чаще.

Каждая версия осмыслена с точки зрения бизнеса.

Преимущества:

* бизнес получает прибыль за счет каждой новой версии
* ранняя обратная связь от пользователей

# Рефакторинг

Невозможно сразу написать идеальный код. Очень часто над делать рефакторинг с целью его улучшения.

Рефакторинг есть следствие эволюции системы, ее естественного развития

Сравнить с ремонтом. Можно забыть о подтекающем кране, о сломавшейся форточке, о плитке, которая отходит. Ремонт каждого из этих поломок длиться полдня. Но в один прекрасный момент кухня дойдет до такого состояния, что в ней невозможно будет находиться. И тогда вы отколупаете всю плитку, которая и так на ладан дышит, снимите окна, из которых стекла вылетают. Ремонт будет длиться неделю, и при этом на кухне будет невозможно находиться.

Но мы не можем изменять систему каждый раз, когда это требуется без наличия тестов.

# Тестирование

Как обычно происходит создание системы?

Мы проектируем систему, создаем ее, тщательно тестируем и отдаем заказчику.

А вот так происходит разработка в XP. Мы начинаем с тестов. Это обязательное правильно.

Все, что не может быть измерено не существует.

Возражения:

* я не тестировщик
* занимает слишком много времени
* если я меняю класс, мне приходится менять и тест
* долго ждать пока пройдут все тесты
* я не знаю как протестировать
* все работает, тесты не нужны

Можно проверить работоспособность своей системы за считанные секунды.

Тесты должны:

* быть независимыми друг от друга
* проходить быстро

Написание тестов:

* защищает код от энтропии
* облегчает модификацию системы, придает уверенности
* создает самую актуальную документацию
* экономит время, а значит, ускоряет разработку. Скорость – это деньги.

# Парное программирование

В прямом смысле этого слова. Два программиста за одним компьютером.

К этой практике привыкают сложнее всего.

Программирование в паре снижает риски:

* постоянная перепроверка кода улучшает качество
* программисты более продуктивно используют свое время
* устранение зависимости от героев
* взаимное обучение членов команды

Подводные камни:

* нужны частые паузы
* эффект глядящего через плечо
* уделяйте время формированию общего словаря. Например, названия шаблонов проектирования, бизнес термины, объекты разрабатываемой системы
* не затягивайте дискуссии, посвященные дизайну
* единая среда разработки. Идея «горячего места»

Почему я не хочу программировать в паре:

* Я хочу стать героем
* Боюсь продемонстрировать свои недостатки
* Я не уверен в том, что моя идея работает
* У меня плохое настроение
* Я люблю чувство медитации

Как бороться с сопротивлением:

* Не обязательно явно говорить о том, что с этого дня вы работаете только в парах

# Постоянная интеграция

# Простой дизайн

Простой не значит плохой или некачественный.

# Заказчик в команде

# Приемочные тесты

# Метафора системы

# Коллективное владение кодом

Автор кода – вся команда

# Стандарты кодирования

# 40 часовая рабочая неделя

Мы хотим быть свежими и энергичными каждое утро, равно как и уставшими и удовлетворенными каждый вечер. Уставшие разработчики допускают больше ошибок, они не довольны, они увольняются

# Заключение

Если у вас будут эти навыки, вы будете ценными сотрудниками.