

# Программная инженерия, лекция 11 Extreme Programming и Rapid Application Development

Пожидаев М. С.

28 ноября 2022 г.

# Extreme Programming

## *Пример гибкой методологии*

Автор методики Extreme Programming (XP) — американский разработчик Кент Бек. В конце 90-х годов он руководил проектом Chrysler Comprehensive Compensation System, в котором применил практики Extreme Programming. Свою идею подробно изложил в книге «Extreme Programming Explained» 1999 года. К становлению методологии также причастны Уорд Каннингем и Мартин Фаулер.

# ОСНОВЫ

## *Цель, процессы и ценности XP*

Подходит для сложных и неопределенных проектов.

Цель — справиться с постоянно меняющимися требованиями к программному продукту и повысить качество.

Процессы:

кодирование, тестирование, дизайн и слушание.

Ценности:

простота, коммуникация, обратная связь, смелость и уважение.

# Ценности

## *Простота, коммуникация и обратная связь*

### Простота

Разработка начинается с самого простого решения, которое удовлетворит текущую потребность. Члены команды учитывают только то, что должно быть сделано сейчас, и не закладывают функциональность, которая понадобится завтра.

### Коммуникация

Коммуникация между разработчиками ведется не посредством документации, а вживую. Команда активно общается между собой и с заказчиком.

### Обратная связь:

отзыв системы при постоянном тестировании модулей; отзыв заказчика, т.к. он входит в команду и участвует в написании приемочных тестов; отзыв команды во время планирования касательно времени на разработку.

# Ценности

## Смелость

### Смелость

Некоторые методики экстремального программирования настолько непривычны, что требуют смелости и постоянного контроля над собой.

### Уважение

В экстремальном программировании уважение рассматривается с точки зрения уважения к команде и самоуважения. Члены команды не должны заливать изменения, которые ломают компиляцию, модульные тесты, или замедляют работу коллег. Каждый стремится к высшему качеству кода и дизайна.

# Ключевые практики

## Разработка, основанная на тестировании

В XP тесты пишутся самими программистами, причем *ДО написания кода*, который нужно протестировать. Когда пара программистов публикуют код, сразу запускаются модульные тесты. И ВСЕ они должны сработать.

## Парное программирование

Два разработчика работают за одним компьютером над одним куском функциональности продукта. «Одна голова хорошо, а две лучше». Получаем код, в котором хорошо разбираются сразу двое разработчиков.

# Практики

- ▶ *Вся команда.* Все участники проекта с применением XP работают как одна команда. В нее обязательно входит представитель заказчика.
- ▶ *Игра в планирование.* Планирование в XP проводят в два этапа: планирование релиза и планирование итераций. Если команда не успевает выполнить все задачи к дате релиза, то релиз не отодвигается, а режется
- ▶ *Частые релизы.* Версии выпускаются часто, но с небольшим функционалом.
- ▶ *Пользовательские тесты.* Заказчик сам определяет автоматизированные приемочные тесты, чтобы проверить работоспособность очередной функции продукта.

# Практики (продолжение)

- ▶ *Коллективное владение кодом.* Любой разработчик может править любой кусок кода, т. к. код не закреплен за своим автором.
- ▶ *Непрерывная интеграция кода.* Новые части кода сразу же встраиваются в систему — команды ХР заливают новый билд каждые несколько часов.
- ▶ *Стандарты кодирования.* Когда кодом владеют все, важно принять единые стандарты оформления, чтобы код выглядел так, как будто он написан одним профессионалом.
- ▶ *Метафора системы.* Метафора системы — это ее сравнение с чем-то знакомым, чтобы сформировать у команды общее видение.



# Практики (продолжение)

- ▶ *Устойчивый темп.* Команды работают на максимуме продуктивности, сохраняя устойчивый темп. При этом XP негативно относится к переработкам.
- ▶ *Простой дизайн.* Простой дизайн означает делать только то, что нужно сейчас, не пытаясь угадать будущую функциональность.
- ▶ *Рефакторинг.* Рефакторинг — это процесс постоянного улучшения дизайна системы, чтобы привести его в соответствие новым требованиям.

# Преимущества XP

1. Заказчик получает именно тот продукт, который ему нужен.
2. Команда быстро вносит изменения в код.
3. Код всегда работает за счет постоянного тестирования и непрерывной интеграции.
4. Быстрый темп разработки за счет парного программирования, отсутствия переработок, присутствия заказчика.
5. Уход/приход члена команды не разрушит процесс.
6. Затраты на разработку ниже, т. к. команда ориентирована на код, а не на документацию и собрания.

# Недостатки XP

1. Успех проекта зависит от вовлеченности заказчика, которого не так просто добиться.
2. Трудно предугадать затраты времени на проект, т. к. в начале никто не знает всех требований.
3. Успех сильно зависит от уровня программистов.
4. Менеджмент негативно относится к парному программированию.
5. Регулярные встречи с программистами дорого обходятся заказчикам.
6. Из-за недостатка структуры и документации не подходит для крупных проектов.
7. Поскольку гибкие методологии функционально-ориентированы, нефункциональные требования не учитываются.

# RAD

## *Rapid Application Development*

RAD — модель быстрой разработки приложений, ориентированная на скорость и удобство написания кода. Первая версия была создана Барри Боэм в 1986 г., который назвал её «спиральная модель». Каждый виток спирали разбит на четыре сектора. С каждым новым витком идёт углубление и уточнение целей.

Используя идеи Барри, во время своей работы в IBM британец Джеймс Мартин разработал свою систему RAD и окончательно сформулировал их в книге «Быстрая разработка приложений» в 1991 г.

# ОСНОВЫ

## Цели и принципы

Цели указывают на обеспечение основных преимуществ методики быстрой разработки:

повышенная скорость разработки, низкая стоимость, высокое качество.

С последним пунктом есть проблемы, потому что разработчик и заказчик видят предмет разработки по-разному.

Принципы:

минимизация потраченного времени — инструментарий нацелен на уменьшение времени разработки, прототипирование, цикличность разработки, сотрудничество, итерационный подход, комбинирование тестирования и разработки.

# Фазы

## Жизненный цикл в RAD

1. *Анализ и планирование требований.* Определяются требования, функции приложения и их приоритетность. Фаза выполняется преимущественно пользователями при участии разработчиков.
2. *Проектирование.* Часть пользователей участвует в техническом проектировании под руководством разработчиков. JAD (Joint Application Development) — концепция совместной разработки. создаются общая информационная модель приложения, функциональные модели системы и подсистем, рабочие прототипы UI.
3. *Построение.* Выполняется разработка на основе полученных на предыдущих фазах результатах. Пользователи продолжают участвовать в развитии.
4. *Внедрение.* Охватывает обучение пользователей, тестирование и замену старой системы на новую.

# Преимущества RAD

1. *Высокое качество.* Взаимодействие пользователей с прототипами повышает функциональность проектов.
2. *Контроль рисков.* Использование RAD позволяет уже на ранних стадиях сосредоточиться на главных факторах риска и приспособиться к ним. Боэм охарактеризовал спиральную модель как основанную на риске.
3. *Бюджетная эффективность.* за единицу времени выполняется больше проектов в рамках бюджета. в RAD вся критическая для успеха проекта информация раскрывается раньше.

# Недостатки RAD

1. *Риск новизны.* Для большинства ИТ-компаний RAD является диковинкой, требующей переосмысления привычных методик работы.
2. *Постоянное присутствие заказчика.* Если пользователи не вовлечены в разработку на протяжении всего жизненного цикла, это может негативно повлиять на конечный продукт.
3. *Уменьшенный контроль.* Адаптивность процесса как преимущество RAD предполагает выбор между гибкостью и контролем. Повышение гибкости будет понижать контроль.
4. *Скучный дизайн.* Фокусирование на прототипах нередко исключает системный предварительный анализ.
5. *отсутствие масштабируемости.* Используется маленькими и средними командами.



Спасибо за внимание!

Всё о курсе: <https://marigostra.ru/materials/engineering.html>

E-mail: [msp@luwrain.org](mailto:msp@luwrain.org)

Канал в Телеграм: <https://t.me/MarigostraRu>