Технологии разработки программного обеспечения

2017 / 2018, 1 курс, 2 семестр

Пудов Сергей Григорьевич

Лекция 9

- Agile
  - XP
  - Scrum

Agile

Гибкая методология разработки (англ. *Agile software development*, *agile-методы*) — серия подходов к разработке программного обеспечения, ориентированных на использование итеративной разработки, динамическое формирование требований и обеспечение их реализации в результате постоянного взаимодействия внутри самоорганизующихся рабочих групп, состоящих из специалистов различного профиля. Существует несколько методик, относящихся к классу гибких методологий разработки, в частности экстремальное программирование, DSDM, Scrum, FDD.

Большинство гибких методологий нацелены на минимизацию рисков путём сведения разработки к серии коротких циклов, называемых итерациями, которые обычно длятся две-три недели. Каждая итерация сама по себе выглядит как программный проект в миниатюре и включает все задачи, необходимые для выдачи мини-прироста по функциональности: планирование, анализ требований, проектирование, программирование, тестирование и документирование.

Agile Manifesto

Манифест гибкой разработки программного обеспечения (англ. *Agile Manifesto*) — основной документ, содержащий описание ценностей и принципов гибкой разработки программного обеспечения, разработанный в феврале 2001 года на встрече 17 независимых практиков нескольких методик программирования, именующих себя «Agile Alliance».

Мы постоянно открываем для себя более совершенные методы разработки программного обеспечения, занимаясь разработкой непосредственно и помогая в этом другим. Благодаря проделанной работе мы смогли осознать, что:

- **Люди и взаимодействие** важнее процессов и инструментов
- **Работающий продукт** важнее исчерпывающей документации
- **Сотрудничество с заказчиком** важнее согласования условий контракта
- **Готовность к изменениям** важнее следования первоначальному плану

То есть, не отрицая важности того, что справа, мы всё-таки больше ценим то, что слева.

Принципы Agile-манифеста

- Наивысшим приоритетом для нас является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения.
- Изменение требований приветствуется, даже на поздних стадиях разработки. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.
- Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.
- На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.
- Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им.

Принципы Agile-манифеста

- Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.
- Работающий продукт — основной показатель прогресса.
- Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно. Agile помогает наладить такой устойчивый процесс разработки.
- Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.
- Простота — искусство минимизации лишней работы — крайне необходима.
- Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.
- Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Extreme Programming (XP): принципы

Название методологии исходит из идеи применить полезные традиционные методы и практики разработки программного обеспечения, подняв их на новый «экстремальный» уровень.

Принципы XP

Итеративность. Разработка ведется короткими итерациями при наличии активной взаимосвязи с заказчиком. Итерации как таковые предлагается делать короткими, рекомендуемая длительность – 2-3 недели и не более 1 месяца. За одну итерацию группа программистов обязана реализовать несколько свойств системы, каждое из которых описывается в пользовательской истории. Пользовательские истории (ПИ) в данном случае являются начальной информацией, на основании которой создается модуль. Они отличаются от вариантов использования (ВИ). Описание ПИ короткое – 1-2 абзаца, тогда как ВИ обычно описываются достаточно подробно, с основным и альтернативными потоками, и дополняются моделью. ПИ пишутся самими пользователями, которые в XP являются частью команды, в отличие от ВИ, которые описывает системный аналитик. Отсутствие формализации описания входных данных проекта в XP стремятся компенсировать за счет активного включения в процесс разработки заказчика как полноправного члена команды.

Extreme Programming (XP): принципы

- **Простота решений.** Принимается первое простейшее рабочее решение. Экстремальность метода связана с высокой степенью риска решения, обусловленного поверхностностью анализа и жестким временным графиком. Реализуется минимальный набор главных функций системы на первой и каждой последующей итерации; функциональность расширяется на каждой итерации.
- **Интенсивная разработка малыми группами** (не больше 10 человек) **и парное программирование** (когда два программиста вместе создают код на одном общем рабочем месте), активное общение в группе и между группами. Все это нацелено на как можно более раннее обнаружение проблем (как ошибок, так и срыва сроков). Парное программирование направлено на решение задачи стабилизации проекта. При применении XP методологии высок риск потери кода по причине ухода программиста, не выдержавшего интенсивного графика работы. В этом случае второй программист из пары играет роль «наследника» кода. Немаловажно и то, как именно распределены группы в рабочем пространстве – в XP используется открытое рабочее пространство, которое предполагает быстрый и свободный доступ всех ко всем.
- **Обратная связь с заказчиком**, представитель которого фактически вовлечен в процесс разработки.
- **Достаточная степень смелости** и желание идти на риск.

Extreme Programming (XP)

Практики XP (***Лилия Хаф, http://compress.ru/article.aspx?id=12147***)

- **Планирование** процесса (planning game). Вся команда собирается вместе, принимается коллективное решение о том, какие свойства системы будут реализованы в ближайшей итерации. Набор свойств определяется пользовательскими историями. XP-трудоемкость каждого свойства определяется самими программистами.
- Тесное взаимодействие с **заказчиком** (feed-back, on-site customer). Заказчик должен быть членом XP-команды (on-site customer). Он пишет пользовательские истории, выбирает истории, которые будут реализованы в конкретной итерации, и отвечает на вопросы, касающиеся бизнеса. Заказчик должен быть экспертом в автоматизируемой предметной области. Необходимо постоянное наличие обратной связи с заказчиком (feed-back).

Extreme Programming (XP)

- **Метафора** системы (system metaphor). Хорошая метафора системы означает простоту именования классов и переменных. В реальной жизни поиск метафоры — крайне сложное занятие; найти хорошую метафору непросто. В любом случае команда должна иметь единые правила именования.
- **Простая** архитектура (simple design). Любое свойство системы должно быть реализовано как можно проще. Программисты в XP-команде работают под девизом: «Ничего лишнего!». Принимается первое наипростейшее работающее решение, реализуется необходимый уровень функциональности на данный момент. Тем самым экономится время программиста.
- **Стандарты** кодирования (coding conventions). Стандарты кодирования нужны для обеспечения других практик: коллективного владения кодом, парного программирования и рефакторинга. Без единого стандарта выполнять эти практики как минимум сложнее, а в реальности вообще невозможно: группа будет работать в режиме постоянной нехватки времени. Детальные стандарты не требуются, необходимо стандартизировать только важные вещи. Определение наиболее важных объектов стандартизации в XP субъективно.

Extreme Programming (XP)

- **Рефакторинг** (refactoring). Рефакторинг — это оптимизация существующего кода в сторону упрощения, что предусматривает постоянную работу по упрощению кода. Сохраняя код прозрачным и определяя его элементы всего один раз, программисты сокращают число ошибок, которые впоследствии придется устранять. При реализации каждого нового свойства системы программист должен подумать над тем, можно ли упростить существующий код и как это поможет реализовать новое свойство. Кроме того, нельзя совмещать рефакторинг с дизайном: если создается новый код, рефакторинг надо отложить.
- **Парное программирование** (pair programming) — одна из самых известных XP-практик. Все программисты должны работать в парах: один пишет код, другой смотрит. Таким образом, необходимо размещать группу программистов в одном месте, что легче всего сделать на территории заказчика (все необходимые члены команды географически находятся в одном месте); XP наиболее успешно работает в нераспределенных коллективах программистов и пользователей.

Extreme Programming (XP)

- **40-часовая** рабочая неделя. Программист не должен работать более 8 часов в день. Необходимость сверхурочной работы (overtime) — это четкий индикатор проблемы на данном конкретном направлении разработки; к тому же заказчик не платит за сверхурочную работу в XP. Поиск причин сверхурочной работы и их скорейшее устранение — одно из основных правил.
- **Коллективное** владение кодом (collective code ownership). Каждый программист в коллективе XP должен иметь доступ к коду любой части системы и вносить изменения в любой код. Обязательное правило: если программист внес изменения и система после этого работает некорректно, то именно этот программист должен исправить ошибки. В противном случае работа системы уподобится тотальному хаосу.
- **Частая смена версий** (small releases). Минимальная итерация — один день, максимальная — месяц; чем чаще осуществляются релизы, тем больше недостатков системы будет выявлено. Первые релизы помогают выявить недостатки на самых ранних стадиях, далее функциональность системы расширяется (на основании тех же пользовательских историй).

Extreme Programming (XP)

- **Непрерывная интеграция** (continuous integration). Интеграция новых частей системы должна происходить как можно чаще, как минимум раз в несколько часов. Основное правило интеграции следующее: интеграцию можно производить, если все тесты проходят успешно. Если тесты не проходят, то программист должен либо внести исправления и тогда интегрировать составные части системы, либо вообще не интегрировать их. Правило это жесткое и однозначное — если в созданной части системы имеется хотя бы одна ошибка, то интеграцию производить нельзя.
- **Тестирование** (testing). В отличие от большинства остальных методологий тестирование в XP — одно из важнейших составляющих. Экстремальный подход заключается в том, что тесты пишутся до написания кода. Каждый модуль обязан иметь unit test — тест данного модуля; таким образом, в XP осуществляется regression testing (возвратное тестирование, «неухудшение качества» при добавлении функциональности). Большинство ошибок исправляются на стадии кодирования. Еще один важный принцип: тест определяет код, а не наоборот (такой подход носит название test-driven development), то есть кусок кода кладется в хранилище тогда и только тогда, когда все тесты прошли успешно, в противном случае данное изменение кода отвергается.

Продолжение в следующей лекции…