Технология разработки программного обеспечения 2015-2016 гг.

Лекции 3-4: гибкие технологии

Пудов Сергей Григорьевич

Составление ТЗ и плана работ

Техническое задание (ТЗ) — документ, содержащий набор требований к проекту. По итогу составления ТЗ у заказчика и исполнителя должно сформироваться общее видение проекта. Приемочное тестирование продукта будет выполняться с учетом заявленного в ТЗ функционала.

В зависимости от специфики компании и проекта ТЗ может принимать различную форму. По мере развития проекта ТЗ может уточняться в силу изменчивости требований.

На первой итерации ТЗ должно включать следующие пункты.

1. Функционал проекта. Описание с точки зрения пользователя: какие задачи решает продукт, какие покрывает сценарии использования.
2. Формат входных данных.
3. Интерфейс приложения. В каком режиме работает приложение (интерактивный или нет, фоновый процесс, сетевой сервис и т. д.). Какие элементы интерфейса предусмотрены, их поведение.
4. Если приложение принимает аргументы командной строки, то их формат и описание.
5. Если предполагается использование конфигурационного файла, то описание его формата.

Гибкие технологии разработки ПО

Для небольших команд (до 10 участников) альтернативой строго формализованных подходов к разработке ПО являются гибкие (agile) методологии. Гибкие методологии ориентированы на профессионалов, которые мотивированы на создание качественного программного продукта в кратчайшие сроки. Основными положениями гибкого подхода к созданию ПО являются:

- люди и взаимодействие важнее процессов и программных средств;
- работающее ПО важнее исчерпывающей документации;
- взаимодействие с заказчиком важнее согласования условий контакта;
- готовность к изменениям важнее следования первоначальному плану.

Гибкие методологии ориентированы на минимизацию рисков, реализуя короткие итерации длительностью в одну или две недели. Каждая итерация заканчивается выпуском заданной функциональности программного проекта, и реализует этапы работ по планированию, анализу требований, проектированию, кодированию, тестированию и документирование. Отдельная итерация, как правило, недостаточна для выпуска новой версии продукта, но подразумевается что гибкий программный проект готов к выпуску в конце каждой итерации. По окончании каждой итерации, команда выполняет переоценку приоритетов разработки.

Гибкие технологии разработки ПО

Для методологии гибкой разработки декларированы ключевые постулаты, позволяющие командам достигать высокой производительности:

- люди и их взаимодействие;
- доставка работающего программного обеспечения;
- сотрудничество с заказчиком;
- реакция на изменение.

Люди и взаимодействие. Люди - важнейшая составная часть успеха. Отдельные члены команды и хорошие коммуникации важны для высокопроизводительных команд. Для содействия коммуникации гибкие методы предполагают частые обсуждения результатов работы и внесение изменений в решения.

Работающее программное обеспечение важнее всеобъемлющей документации. Все гибкие методологии выделяют необходимость доставки заказчику небольших фрагментов работающего программного обеспечения через заданные интервалы. Программное обеспечение, как правило, должно пройти уровень модульного тестирования, тестирования на уровне системы. При этом объем документации должен быть минимальным. В процессе проектирования команда должна поддерживать в актуальном состоянии короткий документ, содержащий обоснования решения и описание структуры.

Гибкие технологии разработки ПО

Сотрудничество с заказчиком важнее формальных договоренностей по контракту. Чтобы проект успешно завершился, необходимо регулярное и частое общение с заказчиком. Заказчик должен регулярно участвовать в обсуждении принимаемых решений по программному обеспечению, высказывать свои пожелания и замечания. Вовлечение заказчика в процесс разработки программного обеспечения необходимо создания качественного продукта.

Оперативное реагирование на изменения важнее следования плану. Способность реагирования на изменения во многом определяет успех программного проекта. В процессе создания программного продукта очень часто изменяются требования заказчика. Заказчики очень часто точно не знают, чего хотят, до тех пор, пока не увидят работающее программное обеспечение. Гибкие методологии ищут обратную связь от заказчиков в процессе создания программного продукта. Оперативное реагирование на изменения необходимо для создания продукта, который удовлетворит заказчика и обеспечит ценность для бизнеса.

XP – экстремальное программирование

- Короткий цикл обратной связи (Fine-scale feedback)
  - Разработка через тестирование (Test-driven development)
  - Игра в планирование (Planning game)
  - Заказчик всегда рядом (Whole team, Onsite customer)
  - Парное программирование (Pair programming)
- Непрерывный, а не пакетный процесс
  - Непрерывная интеграция (Continuous integration)
  - Рефакторинг (Design improvement, Refactoring)
  - Частые небольшие релизы (Small releases)
- Понимание, разделяемое всеми
  - Простота (Simple design)
  - Метафора системы (System metaphor)
  - Коллективное владение кодом (Collective code ownership) или выбранными шаблонами проектирования (Collective patterns ownership)
  - Стандарт кодирования (Coding standard or Coding conventions)
- Социальная защищённость программиста (Programmer welfare):
  - 40-часовая рабочая неделя (Sustainable pace, Forty-hour week)

XP – экстремальное программирование

Тестирование. XP предполагает написание автоматических тестов (программный код, написанный специально для того, чтобы тестировать логику другого программного кода). Особое внимание уделяется двум разновидностям тестирования:

- юнит-тестирование модулей;
- функциональное тестирование.

Разработчик не может быть уверен в правильности написанного им кода до тех пор, пока не сработают абсолютно все тесты модулей разрабатываемой им системы. Тесты модулей (юнит-тесты) позволяют разработчикам убедиться в том, что каждый из них по отдельности работает корректно. Они также помогают другим разработчикам понять, зачем нужен тот или иной фрагмент кода, и как он функционирует — в ходе изучения кода тестов логика работы тестируемого кода становится понятной, так как видно, как он должен использоваться. Тесты модулей также позволяют разработчику без каких-либо опасений выполнять рефакторинг (refactoring).

XP – экстремальное программирование

Игра в планирование. Как и любая другая игра, планирование имеет своих участников и свою цель.

Ключевой фигурой является, конечно же, заказчик.

Именно он сообщает о необходимости той или иной функциональности. Программисты же дают ориентировочную оценку каждой функциональности. Прелесть игры в планирование заключается в единстве цели и солидарности разработчика и заказчика: в случае победы побеждают все, в случае поражения все проигрывают. Но при этом каждый участник идет к победе своей дорогой: заказчик выбирает наиболее важные задачи в соответствии с бюджетом, а программист оценивает задачи в соответствии со своими возможностями по их реализации.

XP – экстремальное программирование

Заказчик на рабочей площадке.

Основной проблемой разработки программного обеспечения является недостаток знаний программистов в разрабатываемой предметной области.

User Story. User Story (что-то типа рассказа пользователя) — это описание того как система должна работать. Каждая User ***Story*** написана на карточке и представляет какой-то кусок функциональности системы, имеющий логический смысл с точки зрения Заказчика. Форма один-два абзаца текста понятного пользователю (не сильно технического).
User Story пишется Заказчиком. Они похожи на сценарии использования системы, но не ограничиваются пользовательским интерфейсом. По каждой истории пишутся функциональные тесты, подтверждающие что данная история корректно реализована — их еще называют приемочными (Acceptance tests).

XP – экстремальное программирование

Парное программирование.

Весь код для продукционной системы пишется парами. Два разработчика сидят рядом. Один набирает, другой смотрит. Время от времени они меняются. Не разрешается работать в одиночку. Если по какой-то причине второй из пары пропустил что-то (болел, отходил и т.п.) он обязан просмотреть все изменения сделанные первым.

Действует принцип «Одна голова хорошо, а две лучше». Пары обычно находят ***более оптимальные решения***. Кроме того, существенно увеличивается ***качество кода***, уменьшается число ошибок и ускоряется ***обмен знаниями*** между разработчиками. Пока один человек сосредоточивает усилия на стратегическом представлении об объекте, второй реализует его свойства и методы.

XP – экстремальное программирование

Частая интеграция

Разработчики, по возможности, должны интегрировать и выпускать свой код каждые несколько часов. В любом случае никогда нельзя держать изменения дольше одного дня. Частая интеграция позволяет избежать отчуждения и фрагментирования в разработке, когда разработчики не могут общаться в смысле обмена идеями или повторного использования кода. Каждый должен работать с самой последней версией.

Каждая пара разработчиков должна отдавать свой код, как только для этого появляется разумная возможность. Это может быть, когда все UnitTest-ы проходят на 100%. Отдавая изменения несколько раз в день, Вы сводите проблемы интеграции практически к нулю.

XP – экстремальное программирование

Рефакторинг.

Это оптимизация существующего кода с целью его упрощения, Такая работа должна вестись постоянно. 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 следует отложить.

XP – экстремальное программирование

Небольшие релизы.

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 ПИ. Поскольку пользователь включается в процесс разработки начиная с первого релиза, то он оценивает систему и выдает пользовательскую историю и замечания. На основании этого определяется следующая итерация, то есть, каким будет новый релиз. В XP все направлено на обеспечение непрерывной обратной связи с пользователями.

XP – экстремальное программирование

Простота проектирования

XP исходит из того, что в процессе работы условия задачи могут неоднократно измениться, а значит, разрабатываемый продукт не следует проектировать заблаговременно целиком и полностью. XP предполагает, что проектирование — это настолько важный процесс, что его необходимо выполнять постоянно в течение всего времени работы над проектом.

Проектирование должно выполняться небольшими этапами, с учётом постоянно изменяющихся требований. В каждый момент времени следует пытаться использовать наиболее простой дизайн, который подходит для решения текущей задачи, и менять его по мере того, как условия задачи меняются.

XP – экстремальное программирование

Система метафор

Выбор системы метафор нужен, чтобы удержать команду в одних и тех же рамках при именовании классов и методов. То, как вы называете свои объекты, очень важно для понимания общего дизайна системы и повторного использования кодов. Если разработчик в состоянии правильно предугадать, как может быть назван существующий объект, это ведет к экономии времени. Используйте такую систему имен для ваших объектов, которую каждый сможет понять без специфических знаний о системе.

XP – экстремальное программирование

Коллективное владение кодом стимулирует разработчиков подавать идеи для всех частей проекта, а не только для своих модулей. Любой разработчик может изменять любой код для расширения функциональности и исправления ошибок.
С первого взгляда это выглядит как хаос. Однако, принимая во внимание, что как минимум любой код создан парой разработчиков, что тесты позволяют проверить корректность внесенных изменений и что в реальной жизни все равно так или иначе приходится разбираться в чужом коде, становится ясно, что коллективное владение кодом значительно упрощает внесение изменений и снижает риск, связанный с высокой специализацией того или иного члена команды.

XP – экстремальное программирование

Соглашение о кодировании

Люди приходят и уходят. Никто не кодирует в одиночку и код принадлежит всем. Всегда будут моменты, когда необходимо будет понять и скорректировать чужой код. Разработчики будут удалять или изменять дублирующий код, анализировать и улучшать чужие классы и т.п. Следовательно, все должны подчиняться общим стандартам кодирования — форматирование кода, именование классов, переменных, констант, стиль комментариев. Вышесказанное означает, что все члены команды должны договориться об общих стандартах кодирования. Неважно каких. Правило заключается в том, что все им подчиняются. Те, кто не желает их соблюдать покидает команду

XP – экстремальное программирование

Не более чем правила (***just rules***). Члены коллектива, работающего по технологии экстремального программирования, обязуются выполнять изложенные правила. Однако это не более чем правила, и команда может в любой момент изменить их, если ее члены достигнут принципиального соглашения по поводу внесенных изменений. Данный принцип серьезно зависит от человеческого фактора; нарушение дисциплины разработки влечет за собой срывы сроков и в результате ведет к краху проекта.

В итоге мы получаем метод, потенциально обладающий высокой адаптацией к серьезно и часто изменяющимся требованиям к проекту, но в то же время не свободный от ряда принципиальных недостатков и в очень высокой степени зависимый от человеческого фактора.

Таким образом, результат применения метода экстремального программирования может получиться либо экстремально хорошим, либо экстремально плохим

Литература

1. Технология разработки программного обеспечения : учеб. пособие / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2010. – 267 с. : ил.
2. Академия Microsoft: Технологии командной разработки программного обеспечения информационных систем http://www.intuit.ru/studies/courses/4806/1054/info
3. Введение в программные системы и их разработку http://www.intuit.ru/studies/courses/3632/874/info
4. http://ru.wikipedia.org/wiki/Экстремальное_программирование
5. Ауэр К., Миллер Р. Экстремальное программирование. Постановка процесса с первых шагов и до победного конца СПб.: Питер, 2004. — 368 с.