Московский государственный университет экономики, статистики и информатики

Институт Компьютерных Технологий

Кафедра Математического обеспечения информационных систем и инноватики

**Реферат**

по курсу «Методы программирования(3)»

На тему «Методы экстремального программирования»

Выполнил студент группы ДКБ-301:

Доянов С.А. ДКБ-10042

Руководитель:

проф. Ковалевская Е.В.

Москва 2013 г.

**Игра в планирование.**

Наш мир слишком изменчив и непредсказуем, чтобы полагаться на постоянство ситуации. То же происходит и при разработке программного обеспечения: о редкой системе можно сказать, что ее окончательный вид был заранее известен в деталях еще в самом начале разработки. Обычно у заказчика аппетит приходит во время еды: ему постоянно хочется что-то поменять, что-то улучшить, а что-то вообще выбросить из системы. Это и есть изменчивость требований, которую все так боятся. К счастью, человеку дано умение прогнозировать возможные варианты и, таким образом, держать ситуацию под контролем.

В экстремальном программировании планирование - неотъемлемая часть разработки и то, что планы могут поменяться, учитывается с самого начала. Той точкой опоры, методикой, которая позволяет прогнозировать ситуацию и безболезненно мириться с изменениями, является игра в планирование. В ходе такой игры можно быстро собрать известные требования к системе, оценить и запланировать их разработку в соответствии с приоритетностью.

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.

Экстремальное программирование предполагает, что разработчики в состоянии сами решить, за какой промежуток времени они справятся со своими задачами и кто из них охотнее бы решил одну задачу, а кто другую.

В идеальной ситуации игра в планирование с привлечением заказчика и программиста должна проводиться каждые 3-6 недель, до начала следующей итерации разработки. Это позволяет довольно просто внести коррективы в соответствии с успехами и неудачами предыдущей итерации.

**Тестирование.**

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

* тестирование модулей (unittesting);
* 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, т.к. видно, как он должен использоваться. Тесты модулей также позволяют разработчику без каких-либо опасений выполнять [рефакторинг](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%84%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3) (refactoring).

Функциональные тесты предназначены для тестирования функционирования логики, образуемой взаимодействием нескольких (часто - довольно внушительного размера) частей. Они менее детальны, чем юнит-тесты, но покрывают гораздо больше -т.е., у тестов, которые при своём выполнении затрагивают больший объём кода, шанс обнаружить какое-либо некорректное поведение, очевидно, больше. По этой причине в промышленном программировании написание функциональных тестов нередко имеет больший приоритет, чем написание юнитов.

Для XP более приоритетным является подход, называемый TDD ([TestDrivenDevelopment](http://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D1%87%D0%B5%D1%80%D0%B5%D0%B7_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) — сначала пишется тест, который изначально не проходит (т.к. логики, которую он должен проверять, ещё просто не существует), затем реализуется логика, необходимая для того, чтобы тест прошел. TDD, в некотором смысле, позволяет писать код, более удобный в использовании -- потому что при написании теста, когда логики ещё нет, проще всего позаботиться об удобстве будущей системы.

**Парное программирование.**

*Парное программирование* предполагает, что весь код создается парами программистов, работающих за одним компьютером. Один из них работает непосредственно с текстом программы, другой просматривает его работу и следит за общей картиной происходящего. При необходимости клавиатура свободно передается от одного к другому. В течение работы над проектом пары не фиксированы: рекомендуется их перемешивать, чтобы каждый программист в команде имел хорошее представление о всей системе. Таким образом, парное программирование усиливает взаимодействие внутри команды.

Удалённое (дистанционное) парное программирование,

Такжеизвестное каквиртуальное парное программированиеилираспределённое парное программирование, это разновидность парного программирования, в котором два программиста находятся на расстоянии друг от друга, работая через средство совместного редактирования кода, удалённый рабочий стол, или посредством плагина для интегрированной системы разработки. Удалённое парное программирование добавляет дополнительные трудности, отсутствующие в обычном парном программировании, такие как дополнительные задержки координации, зависимость от более тяжеловесных инструментов управления задачами вместо таких упрощённых вариантов, как карточки с индексами, и отсутствие невербальных средств общения, приводящее к непониманию и конфликтам в таких вопросах, как "чья сейчас клавиатура".

Итак сделаем вывод, что программирование в паре, обладает многочисленными преимуществами, как экономическими, так и многими другими:

- Любые решения в области дизайна принимаются не одной головой, а двумя;

- Какую бы часть системы вы не взяли, в ней будут хорошо разбираться по крайней мере два человека;

- Если разработкой одного участка кода занимаются одновременно два человека, снижается вероятность ошибок, неаккуратного кода, отсутствие необходимых тестов и т.п.;

- Партнеры в парах постоянного меняются, благодаря чему знание о внутреннем устройстве разрабатываемой системы быстро распространяется между членами команды;

- Происходит постоянная перепроверка чужого кода: один партнер пишет код, другой просматривает этот код.

Как подтверждают исследования, программирование в паре на самом деле эффективнее, чем программирование в одиночку. Возможно, когда вы начнете программировать в паре, вам придется смириться с небольшим снижением скорости.

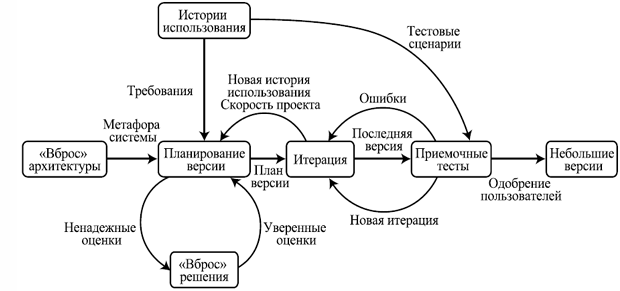
### Заказчик всегда рядом.

Основной проблемой разработки программного обеспечения является недостаток знаний программистов в разрабатываемой предметной области. Экстремальное программирование нашло выход и из этой ситуации. Нет, это не стажировка разработчика на предприятии заказчика - он тогда не захочет программировать. Наоборот, это участие заказчика в процессе разработки.  
Разве может программист, досконально не понимая суть вопроса и не будучи телепатом, угадать, чего хочет заказчик? Ответ очевиден. Самым простым способом преодолеть такое неудобство - а экстремальное программирование учит нас находить самые простые решения - будет задать заказчику прямой вопрос. Более строгие подходы требуют всеобъемлющего предварительного анализа разрабатываемой области. В определенных случаях это оправдано, хотя и дороже обходится. Реальный опыт ведения приземленных проектов показывает, что невозможно собрать все требования заранее. Более того, даже если предположить, что все требования на текущий момент собраны, все равно останется одно узкое место: программы, как и все в природе, не создаются мгновенно, а тем временем бизнес-процессы могут поменяться. Это следуетучитывать.

Многие сомневаются в возможности привлечения заказчика к процессу разработки. Действительно, заказчики бывают разные. Если привлечь заказчика или его представителя не удается, иногда оказывается целесообразным временный наем специалиста в разрабатываемой области. Такой шаг сократит неясности в работе, повысит скорость разработки и приблизит проект к тому, что желает получить заказчик. Это может быть выгодно и с финансовой стороны: ведь оплата труда программиста порой значительно превышает оклад специалистов других отраслей.

**Живое планирование.**

Его задача - как можно быстрее определить объем работ, которые нужно сделать до следующей версии ПО. Решение принимается, в первую очередь, на основе приоритетов заказчика (т.е. его потребностей, того, что нужно ему от системы для более успешного ведения своего бизнеса) и, во вторую, на основе технических оценок (т.е. оценок трудоемкости разработки, совместимости с остальными элементами системы и пр.). Планы изменяются, как только они начинают расходиться с действительностью или пожеланиями заказчика.



**Рис.1** Схема потока работ в XP

**Метафора системы.**

Метафора в достаточно простом и понятном команде виде должна описывать основной механизм работы системы. Это понятие напоминает архитектуру, но должно гораздо проще, всего в виде одной-двух фраз описывать основную суть принятых технических решений.

Архитектура - это некоторое представление о компонентах системы и о том, как они взаимосвязаны между собой. Разработчики используют архитектуру для того, чтобы понять, в каком месте системы добавляется некоторая новая функциональность, и с чем будет взаимодействовать некоторый новый компонент.

Метафора системы (systemmetaphor) - это аналог того, что в большинстве методик называется архитектурой. Метафора системы дает команде представление о том, каким образом система работает в настоящее время, в каких местах добавляются новые компоненты, и какую форму они должны принять.

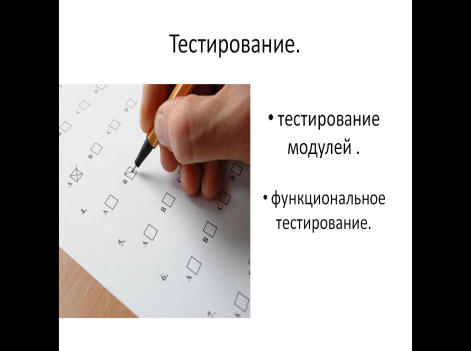
Подобрав хорошую метафору, вы облегчаете команде понимание того, каким образом устроена ваша система. Иногда сделать это непросто.

**Использование кода как средства коммуникации**

Код рассматривается как важнейшее средство общения внутри команды. Ясность кода - один из основных приоритетов. Следование стандартам кодирования, обеспечивающим такую ясность, обязательно. Такие стандарты, помимо ясности кода, должны обеспечивать минимальность выражений (запрет на дублирование кода и информации) и должны быть приняты всеми членами команды.

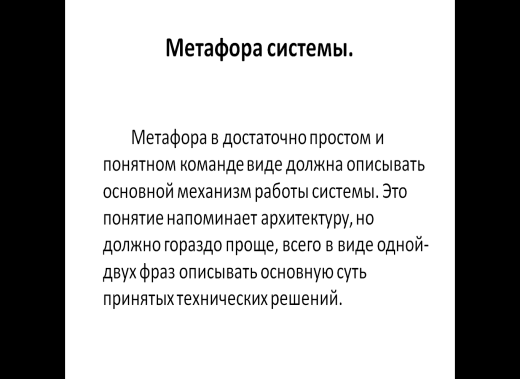


**Глоссарий**

**1.Экстремальное программирование** ( *ExtremeProgramming*, *XP*) — одна из гибких методологий разработки программного обеспечения.

**2.Тестирование-**  определение одной или более характеристики объекта оценки соответствия согласно процедуре (ГОСТ Р ИСО/МЭК 17000).

**3.Плани́рование** — оптимальное распределение ресурсов для достижения поставленных целей, деятельность (совокупность процессов), связанных с постановкой целей (задач) и действий в будущем.

**4.Итерация**  — организация обработки данных, при которой действия повторяются многократно

**5.Парное программирование** — техника программирования, при которой весь исходный код создаётся парами людей, программирующих одну задачу, сидя за одним рабочим местом.

**6.Релиз**-выпуск новой версии

**7.Методология разработки** – серия подходов к разработке ПО.

**8.Программное обеспеччение** -все или часть программ, процедур, правил и соответствующей документации системы обработки информации (*ISO/IEC 2382-1: 1993.Informationtechnology — Vocabulary — Part 1: Fundamentalterms*)

**9.Программирование** -процесс создания компьютерных программ.

**10.Интеграция** - процесс объединения частей в целое

**Список литературы**

1.Кент Бек «Экстремальное программирование», Питер, 2002

2.Кен Ауэр, Рой Миллер «Экстремальное программирование: постановка процесса с первых шагов и до победного конца»,  Питер, 2003

3.Кент Бек «Экстремальное программирование: разработка через тестирование», Питер, 2003

4.Кент Бек, Мартин Фаулер «Экстремальное программирование: планирование»,Питер, 2003

# 5. [Питер Гудлиф](http://www.ozon.ru/context/detail/id/4318341/#tab_person) «Ремесло программиста. Практика написания хорошего кода» [Символ-Плюс](http://www.ozon.ru/context/detail/id/856498/), 2009

6.<http://citforum.ru/SE/project/programing.shtml>статья Александра Федоренко,Издательский Дом "КОМИЗДАТ"

7.<http://www.compress.ru/article.aspx?id=11721&iid=457>статья КомпьютерПресс 9'2000 Георгий Филягин

8.<http://www.cmcons.com/articles/gibkie_metodologii/testirovanie_v_svete_ekstremalnogo_programmirovanija/>статья  Сардарян Рубен, Новичков Александр

9.<http://habrahabr.ru/post/151563/>статья

10.<http://www.xakep.ru/post/28962/>статья журнал «Хакер»,