Приоритетный национальный проект «ОБРАЗОВАНИЕ»

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный университет информационных технологий, механики и оптики»

Д. Г. Шопырин

Рекомендации по самостоятельной работе студентов

Управление проектами разработки ПО. Дисциплина «Гибкие технологии разработки программного обеспечения»

Санкт-Петербург
2007

Содержание

1. Основные принципы гибких технологий разработки программного обеспечения

Цель и задачи темы

Дать представление об основных принципах гибких технологий разработки программного обеспечения. Показать, в чем принципиальное отличие гибких технологий от традиционных методологий разработки программного обеспечения. Описать основные преимущества и недостатки гибких технологий. Описать область применимости гибких технологий при промышленной разработке программного обеспечения.

Требования к уровню освоения темы

Студенты должны иметь общее представление об основных принципах гибких технологий разработки программного обеспечения и понимать в чем состоит отличие от традиционных методологий разработки программного обеспечения.

Рекомендации по самостоятельному изучению темы

Для понимания гибких технологий разработки программного обеспечения необходимо получить достаточно полное представление о *традиционных* моделях процесса разработки ПО. Ниже приведены рекомендации по самостоятельному изучению традиционных моделей:

- Ознакомьтесь с каскадными моделями процесса разработки ПО на основе [12, глава 7]. Принцип каскадных (*waterfall*) моделей процесса разработки ПО стал популярным в 1970-х годах и по-прежнему используется в качестве стандарта в производственной практике. Каскадная модель структурирует фазы процесса разработки ПО в виде линейного каскада этапов. Выходные данные предыдущего этапа поступают на вход другого.

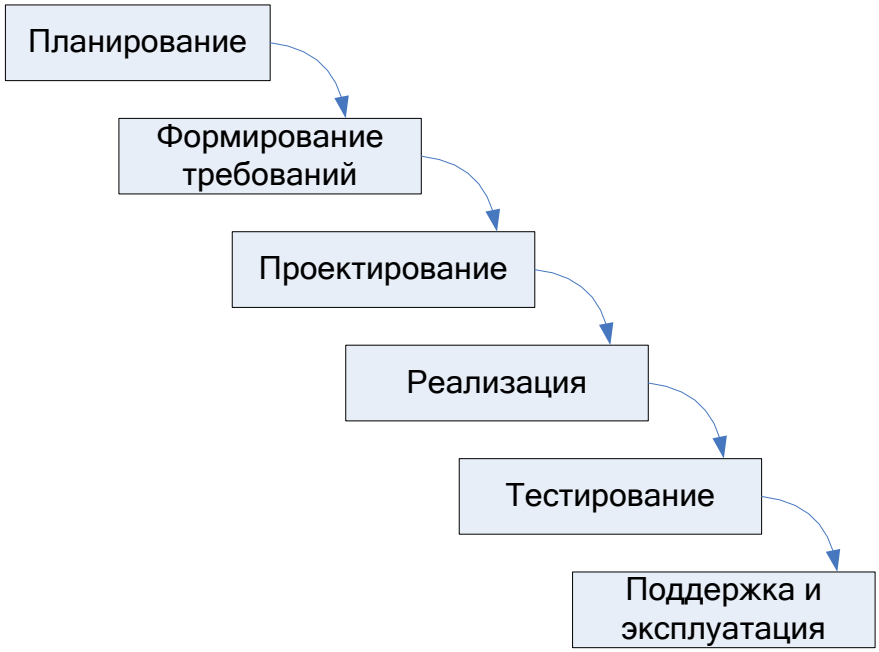

Рис. 1. Каскадная модель разработки ПО

- Ознакомьтесь с примерами практической реализации каскадной модели на основе [12, глава 7]. Существуют разнообразные реализации каскадной модели разработки ПО, оформленные, например, в виде конкретных отраслевых или ведомственных стандартов. Данные стандарты предписывают конкретный состав и порядок действий, которые должны выполняться на каждом этапе жизненного цикла разработки ПО.

- Ознакомьтесь с критикой каскадных моделей на основе [12, глава 7]. Каскадные модели внесли фундаментальный вклад в понимание процессов разработки ПО. Во-первых, каскадные модели утверждают, что процесс разработки ПО должен подчиняться дисциплине, разумному планированию и управлению. Во-вторых, реализация программного продукта должна быть отложена до полного понимания целей этой реализации. Недостатки каскадных моделей связаны с жесткостью и линейностью процесса, что в значительной мере затрудняет и усложняет процесс разработки ПО.

- Ознакомьтесь со спиральной моделью процесса разработки ПО на основе [12, глава 7]. Основной характеристикой спиральной модели является то, что она циклична (в отличии от каскадных моделей). Каждый цикл спирали состоит из четырех стадий, каждая стадия, в свою очередь, представлена одним квадрантом декартовой плоскости. Спиральную модель можно рассматривать как метамодель, потому что она может включать в себя любую другую модель процесса разработки ПО.

Рис. 2. Спиральная модель разработки ПО

- Ознакомьтесь с Rational Unified Process на основе [13]. Rational Unified Process – это итеративный процесс, обеспечивающий большую гибкость при учете новых требований и или тактических изменений в деловых целях, при этом позволяющий проекту заранее идентифицировать и разрешать риски. Rational Unified Process предполагает управление *требованиями* и *изменениями*, что гарантирует общее понимание ожидаемых функциональных возможностей, ожидаемый уровень качества и гарантировать наилучшее управления затратами и графиками выполнения.

- Ознакомьтесь с моделью зрелости процессов разработки ПО (*CMM*) на основе [14]. Ключевым понятием стандарта CMM является зрелость организации. Незрелой считается организация, в которой процесс разработки ПО зависит только от конкретных исполнителей и менеджеров, и решения зачастую просто импровизируются «на ходу» (так называемый *творческий подход*). С другой стороны, в зрелой организации имеются чётко определённые стандарты на процессы разработки, тестирования и внедрения ПО, правила оформления конечного программного кода, компонентов, интерфейсов и т. д. Все это составляет инфраструктуру и корпоративную культуру, поддерживающую процесс разработки программного обеспечения.

- Ознакомьтесь с критикой CMM [15]. Основные проблемы модели CMM состоят в следующем. Модель CMM не имеет формальной теоретической базы (модель основана на «опыте специалистов»). Модель CMM уделяет большое внимание процессам, игнорируя при этом людей. Модель CMM потворствует достижению искусственной цели достижения более высокого уровня зрелости вместо реальной цели улучшения процессов.

- Ознакомьтесь с критикой использования формальных методологий на основе [10, глава 17]. Формальные методологии

навязывают организациям единый шаблон, гарантирующий высокий уровень бюрократии, возможность использовать только ограниченное количество методов, отсутствие ответственности и общую потерю мотивации.

Далее необходимо ознакомится основными принципами гибких технологий разработки программного обеспечения. Для начала желательно ознакомится с манифестом гибких технологий разработки программного обеспечения по материалам сайта http://agilemanifesto.org/.

Также для понимания сути гибких технологий разработки программного обеспечения рекомендуется ознакомиться с основами экстремального программирования по материалам книги [1]. Экстремальное программирование является одной из наиболее известных гибких технологий разработки программного обеспечения. Также желательно ознакомится с другими распространенными методологиями гибкой разработки программного обеспечения по материалам статьи [16].

По итогам ознакомления с традиционными и гибкими технологиями программного обеспечения постарайтесь ответить на следующие вопросы:

- Опишите основные технологические сложности, которые возникают при разработке программного обеспечения.
- Дайте определение понятия «проект».
- Опишите каскадные модели разработки программного обеспечения.
- Опишите основные недостатки каскадных моделей разработки программного обеспечения.
- Опишите спиральную модель разработки программного обеспечения.
- Опишите основные принципы Rational Unified Process.
- Опишите архитектуру процесса разработки программного обеспечения в Rational Unified Process.
- Опишите структуру жизненного цикла процесса разработки программного обеспечения в Rational Unified Process.
- Опишите технологию визуального моделирования, используемую в Rational Unified Process.
- Опишите технологию управления прецедентами, используемую в Rational Unified Process.

- Опишите технологию управления требованиями, используемую в Rational Unified Process.

- Опишите пять уровней зрелости производственного процесса в соответствии с моделью CMM.

- Приведите основные принципы гибких технологий разработки программного обеспечения.

- Опишите итеративный процесс разработки программного обеспечения.

- Опишите основные различия между гибкими и традиционными технологиями разработки программного обеспечения.

- Можно ли сказать, что гибкие технологии отвергают проектирование архитектуры и планирование проекта?

- Какова область применимости гибких технологий разработки программного обеспечения?

Тема семинара: «Место гибких технологий разработки программного обеспечения среди современных технологий программирования»

Темы докладов:

1. Каскадные модели разработки программного обеспечения.

2. Спиральные модели разработки программного обеспечения.

3. Обзор Rational Unified Process.

4. Обзор модели зрелости процессов разработки программного обеспечения CMM.

5. Критика формальных моделей разработки программного обеспечения.

6. Обзор гибких моделей разработки программного обеспечения.

2. Разработка через тестирование

Цель и задачи темы

Дать представление об основных принципах разработки через тестирование и его исключительном влиянии на гибкий процесс разработки программного обеспечения в целом. Описать процесс разработки через тестирование. Научить основным приемам разработки и реализации тестов.

Требования к уровню освоения темы

Студенты должны получить исчерпывающее представление о современных технологиях разработки через тестирование. У студентов должно быть сформировано понимание исключительной важности разработки через тестирование. Студенты должны овладеть основными навыками современных технологий разработки через тестирование.

Разработка через тестирование является краеугольным камнем гибких технологий программирования. Использование большинства других практик гибкой разработки программного обеспечения (например, рефакторинга или непрерывной интеграции) без использования модульных тестов скорее всего окажется неоправданным.

Ниже приведены рекомендации по самостоятельному изучению разработки через тестирование:

- Ознакомьтесь с основами разработки через тестирование по материалам книги [4]. Разработка через тестирование основана на очень простых правилах. Однако полноценное внедрение технологии разработки через тестирование обычно оказывается нетривиальным, так как это связано с преодолением массы организационных и технических трудностей.

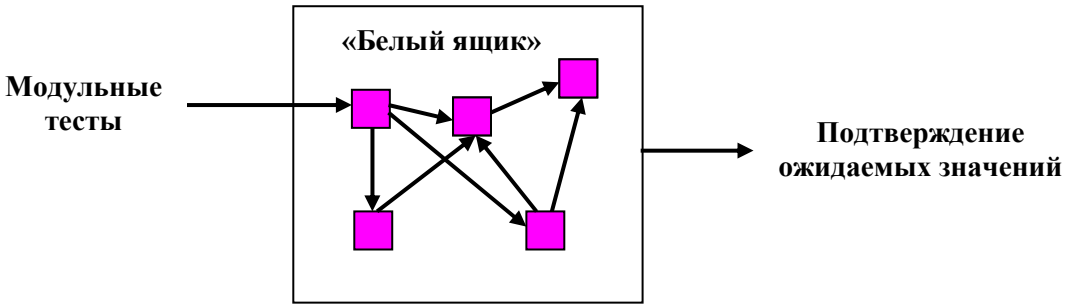

Рис. 3. Модульное тестирование

Рис. 4. Функциональное тестирование

- Ознакомьтесь с фреймворками для написания тестов семейства xUnit по материалам книги [4, часть 2] и сайтов [17, 18]. При изучении фреймворков семейства xUnit обратите внимание на общую семантику реализации модульных тестов, независящую от конкретного языка программирования.

- Ознакомьтесь со средствами группировки тестов во фреймворках семейства xUnit по материалам сайтов [17, 18].

- Ознакомьтесь со способами инициализации тестов во фреймворках xUnit по материалам сайтов [17, 18].

- Ознакомьтесь с моделями утверждений во фреймворках семейства xUnit по материалам сайтов [17, 18].

- Ознакомьтесь со способами ожидания исключений во фреймворках семейства xUnit по материалам сайтов [17, 18].

- Ознакомьтесь с другими фреймворками для написания тестов по материалам сети Интернет [19].

- Ознакомьтесь со средствами интеграции модульного тестирования в современные средства разработки [20, 21].

- Ознакомьтесь с фреймворком тестирования и документации RSpec по материалам сети Интернет [22, 23]. С одной стороны, фреймворк RSpec может рассматриваться как еще один вариант реализации фреймворка для написания модульных тестов. С другой стороны, фреймворк RSpec может рассматриваться как предметно-ориентированный язык программирования для описания ожидаемого поведения системы.

При модульном тестировании современных программных систем часто приходится сталкиваться с тестированием объектов, которые связаны со сложными и тяжеловесными ресурсами. В таких случаях используется технологиях так называемых *поддельных объектов* (*mock objects*). Ниже приведены рекомендации по самостоятельному изучению технологии использования поддельных объектов:

- Ознакомьтесь с технологией использования поддельных объектов по материалам статей [24-26]. Использование поддельных объектов значительно упрощает модульное тестирование нетривиального объектно-ориентированного кода. При этом использование поддельных объектов позволяет избежать засорения кода системы инфраструктурой, необходимой исключительно для модульного тестирования.

- Ознакомьтесь с фреймворками для реализации поддельных объектов по материалам сети Интернет [27-29]. Данные фреймворки позволяют значительно упростить написание модульных тестов с использованием поддельных объектов. Обратите внимание на фреймворки, не нарушающие статическую систему типов современных объектно-ориентированных языков программирования. Использование фреймворков данной категории значительно упрощает последующий рефакторинг кода, оттестированного с помощью поддельных объектов.

При модульном тестировании современных программных систем часто возникают технологические сложности, связанные, например, с модульным тестированием графического интерфейса. Ниже приведены рекомендации по самостоятельному изучению некоторых специализированных технологий модульного тестирования:

- Изучите современные технологии модульного тестирования графического интерфейса пользователя по материалам сети Интернет [30, 31]

- Изучите современные технологии модульного тестирования приложений баз данных по материалам сети Интернет [32].

- Изучите современные технологии модульного тестирования web-приложений по материалам сети Интернет [33].

Для закрепления пройденного материала, разработайте следующие программы:

Модульное тестирование: калькулятор

Используя технологию разработки через тестирование, разработайте программу, основной функциональностью которой является вычисление строковых выражений. На вход приложения подается строка, которая содержит математическое выражение (например, `"(1 + 2) * 3"`). Программа должна выдавать численное значение, являющееся результатом вычисления входного выражения (например, `"9"`). Программа должна поддерживать все основные математические операции, скобки, возведение в произвольную степень и тригонометрические функции. Необходимо полностью протестировать функциональность программы с помощью модульного тестирования.

Тестирование с помощью поддельных объектов: календарь

Используя технологии модульного тестирования с помощью поддельных объектов и модульного тестирования приложений с графическим интерфейсом, разработайте приложение, которое выполняет

основные функции календаря. Календарь должен обладать графическим пользовательским интерфейсом и поддерживать такие операции, как создание напоминаний, создание повторяющихся напоминаний, создание будильников и так далее.

Вариант: разработайте календарь с web-интерфейсом.

Тестирование приложений баз данных: отдел кадров

Используя технологии модульного тестирования приложений баз данных, разработайте приложение, которое выполняет основные функции по управлению персоналом организации. Приложение должно позволять хранить информацию о структуре организации (структура отделов, их руководство и персонал) и выполнять основные операции (прием сотрудника на работу, увольнение сотрудника, перевод сотрудника в другой отдел).

По итогам самостоятельного изучения разработки через тестирование постарайтесь ответить на следующие вопросы:

- Опишите основные принципы разработки через тестирование.
- Опишите типичный цикл разработки через тестирование.
- Опишите влияние разработки через тестирование на сроки проекта.
- Чем отличаются модульные тесты от функциональных тестов?
- Опишите основные средства автоматизации разработки через тестирование.
- Что такое «утверждение» в фреймворках семейства xUnit?
- Что такое «категория теста» в фреймворках семейства xUnit?
- Что такое «поддельный объект»?
- Опишите основные ситуации, когда для модульного тестирования необходимо использовать поддельные объекты.
- Опишите основные технологии тестирования пользовательского интерфейса.
- Опишите основные технологии тестирования приложений баз данных.
- Опишите основные технологии тестирования web-приложений.

Тема семинара: «Разработка через тестирование»

Темы докладов:

1. Разработка через тестирование: основные положения.
2. Обзор фреймворков реализации модульных тестов.
3. Обзор фреймворков реализации поддельных объектов.
4. Обзор фреймворков модульного тестирования графического интерфейса пользователя.
5. Средства автоматизации разработки модульных тестов (интеграция в среду разработки, анализ покрытия кода тестами, тестирование модульных тестов).

3. Кодирование и управление исходным кодом

Цель и задачи темы

Сформировать представление о принципах кодирования, общепринятых в гибких технологиях разработки программного обеспечения. Описать современные технологии управления исходным кодом проекта.

Требования к уровню освоения темы

Студенты должны получить исчерпывающее представление о современных принципах кодирования, общепринятых в гибких технологиях разработки программного обеспечения. Студенты должны получить общее представление о современных технологиях управления исходным кодом.

Культура кодирования и управления исходным кодом оказывают непосредственное влияние на успех проекта в целом. Более того, культура кодирования и управления исходным кодом существенно влияют на возможность использования других практик гибких технологий разработки программного обеспечения, таких как парное программирование, совместное владение кодом и непрерывная интеграция.

Подавляющее большинство современных языков программирования позволяют использовать разнообразные стили кодирования. Беспорядочное смешение нескольких стилей программирования в рамках одного проекта значительно затрудняет внедрение таких практик, как парное программирование, совместное владение кодом и рефакторинг. Поэтому целесообразным является внедрение единого стандарта кодирования, который включает в себя как общие требования по стилю оформления кода, так и требования по общим вопросам программирования. Ниже приведены некоторые рекомендации по самостоятельному ознакомлению с современными стандартами кодирования:

- Ознакомьтесь с основными соглашениями о наименовании по материалам статьи [34]. Конвенция о наименовании – это набор правил, который определяет выбор последовательности символов, использующийся для идентификаторов в исходном коде или документации.

- Ознакомьтесь с распространенными стандартами кодирования по следующим материалам [34-37]. Стандарт кодирования является общим набором правил требований, который определяют написание исходного кода и документирования программной системы.

Важнейшими практиками гибких технологий разработки программного обеспечения являются также парное программирование и совместное

владение кодом. Парное программирование позволяет увеличить дисциплину программирования и значительно повысить качество кода. Совместное владение кодом позволяет осуществлять итеративный рефакторинг системы и значительно снижает риски, связанные с так называемым «фактором грузовика». Ниже приведены некоторые рекомендации по самостоятельному ознакомлению с технологиями парного программирования и коллективного владения кодом:

- Ознакомьтесь с технологиями парного программирования и коллективного владения кодом по материалам [1, глава 16].

- Ознакомьтесь с критикой парного программирования по материалам статей [38].

Гибкие технологии предполагают использование развитых средств управления кодом, таких как системы контроля версий, средства автоматической сборки, средства непрерывной сборки, системы отслеживания ошибок и интегрированные системы управления жизненным циклом приложения.

Системы контроля версий

Современные системы контроля версий используются при разработке программных систем любых масштабов, от небольших студенческих проектов до современных операционных систем. Системы контроля версий позволяют полностью контролировать все изменения, внесенные в исходный код системы. Использование систем контроля версий предполагает некоторое изменение технологического процесса программирования, которое зависит от архитектуры и особенностей используемой системы версий. Ниже приведены некоторые рекомендации по самостоятельному изучению систем контроля версий и способов работы с ними.

- Ознакомьтесь с основными моделями реализации систем контроля версий по материалам книги [39, глава 2].

- Ознакомьтесь с обзором современных систем контроля версий по материалам статьи [40].

- Изучите рекомендуемый жизненный цикл работы с системой контроля версий на примере Subversion (или аналогичной системы) по материалам книги [39, глава 3].

- Изучите рекомендуемый жизненный цикл работы с системой контроля версий на примере Microsoft Visual SourceSafe (или аналогичной системы) по материалам сайта [41].

Средства автоматической сборки

Управление конфигурацией современного продукта зачастую требует значительных трудозатрат. Средства автоматической сборки и управления

конфигурацией проекта позволяют собирать различные версии разрабатываемой системы в полностью автоматическом режиме, что значительно увеличивает производительность команды разработчиков. Ниже приведены некоторые рекомендации по самостоятельному изучению средств автоматической сборки и управления конфигурацией и технологии их использования.

- Ознакомьтесь с современными средствами сборки по материалам сайтом [42-44].

- Изучите рекомендованные способы использования средств автоматической сборки и управления конфигурацией по материалам статей [45, 46].

Системы непрерывной сборки

Системы непрерывной сборки призваны обеспечивать возможность использования технологии непрерывной интеграции системы. Ниже приведены некоторые рекомендации по самостоятельному ознакомлению с современными системами непрерывной сборки:

- Ознакомьтесь с современными системами непрерывной сборки по материалам сайтов [47, 48].

- Ознакомьтесь с рекомендациями по использованию систем непрерывной сборки по материалам статьи [49].

Системы отслеживания ошибок

Системы отслеживания ошибок (*bug/issue tracking*) позволяют отслеживать и учитывать дефекты, которыми обладает разрабатываемая программная система. Использование автоматической системы отслеживания и учета ошибок позволяет значительно повысить качество разрабатываемой системы. Ниже приведены некоторые рекомендации по самостоятельному ознакомлению с современными системами отслеживания ошибок:

- Ознакомьтесь с основами выбора и использования системы отслеживания ошибок по материалам статей [50].

- Ознакомьтесь с обзором современных системам отслеживания ошибок по материалам сайта [51].

Интегрированные системы управления жизненным циклом

Интегрированные системы управления жизненным циклом совмещают и интегрируют все вышеперечисленные системы, такие как система контроля версий, система автоматической сборки, система непрерывной сборки, система отслеживания ошибок. В результате интегрирования всех вышеперечисленных подсистем получается единая интегрированная среда, которая позволяет решать все задачи, связанные с управлением исходным кодом в проектах, которые разрабатываются на основе гибких технологий. Использование интегрированных систем управления жизненным циклом

приложения позволяет значительно повысить производительность команды. Ниже приведены некоторые рекомендации по самостоятельному ознакомлению с современными интегрированными системами управления жизненным циклом приложения:

- Ознакомьтесь с концепцией систем управления жизненным циклом приложения по материалам сайта [52].

Для закрепления пройденного материала, выполните следующие задания:

- Выберите один из стандартов кодирования и используйте его при выполнении лабораторных заданий и курсовых проектов. Выбранный стандарт кодирования должен соответствовать используемому языку программирования и использоваться всеми студентами, участвующими в разработке данного проекта. Обратите внимания на сложности, которые возникают при выборе стандарта и внедрению его среди всех членов команды.

- Воспользуйтесь технологиями парного программирования и коллективного владения кодом при выполнении лабораторных заданий и курсовых проектов. Обратите внимания на выгоды, полученные от использования данных технологий, и на возникшие сложности.

- Воспользуйтесь одной из свободно-распространяемых систем контроля версий (например, Subversion) при выполнении лабораторных заданий и курсовых проектов. Постарайтесь четко следовать рекомендованному циклу работы с выбранной системой контроля версий. Проанализируйте результаты использования системы контроля версий в студенческих проектах.

- Выберите одно из средств автоматической сборки и используйте его при выполнении лабораторных заданий и курсовых проектов. Разработайте план управления конфигурацией проекта. Сборка проекта из исходного кода должна осуществляться в полностью автоматическом режиме. Процесс сборки, как минимум, должен включать компиляцию исходного кода, прогон модульных тестов и упаковку системы (если требуется).

- Выберите одну из систем непрерывной интеграции и используйте ее при выполнении лабораторных заданий и курсовых проектов. Разработайте график выпуска версий приложения.

- Выберите одну из систем отслеживания дефектов и используйте ее при выполнении лабораторных заданий и курсовых проектов.

Разработайте модель жизненного цикла дефекта, подходящую для вашего проекта.

По итогам самостоятельного изучения технологий кодирования и управления исходным кодом постарайтесь ответить на следующие вопросы:

- Приведите основные критерии простоты кода.
- Обоснуйте целесообразность использования единого стандарта программирования.
- Перечислите основные распространенные соглашения о наименовании.
- Опишите достоинства и недостатки использования технологии парного программирования.
- Обоснуйте необходимость использования систем контроля версий.
- Опишите основные модели реализации современных систем контроля версий.
- Что такое тэг (метка) в современных системах контроля версий?
- Что такое стабилизационная ветка в современных системах контроля версий?
- Обоснуйте целесообразность использования автоматической подсистемы управления конфигурацией.
- Обоснуйте целесообразность использования систем непрерывной интеграции.
- Обоснуйте целесообразность использования систем отслеживания ошибок.

Тема семинара: «Кодирование и управление исходным кодом»

Темы докладов:

1. Достоинства и недостатки технологии парного программирования.
2. Стандарты и стиль программирования: соглашения по именованию переменных, классов, методов и форматировании кода.
3. Обзор современных систем контроля версий.

4. Обзор современных средств сборки и непрерывной интеграции.
5. Обзор современных систем отслеживания ошибок.

4. Проектирование и управление требованиями

Цель и задачи темы

Сформировать представление о принципах проектирования и управления требованиями в гибких технологиях разработки программного обеспечения. Описать технологические сложности, возникающие при традиционных подходах к проектированию и управлению требованиями. Обосновать целесообразность итеративного проектирования и управления требованиями. Сформировать представление о рефакторинге и его влиянии на проект в целом. Обосновать экономическую целесообразность итеративного рефакторинга исходного кода системы. Описать современные практики итеративного проектирования и управления требованиями.

Требования к уровню освоения темы

Студенты должны получить общее представление о сложностях, возникающих при использовании традиционных подходов к проектированию и управлению требованиями. Студенты должны иметь полное представление об итеративных методах проектирования и управления требованиями и целесообразности их использования. Студенты должны быть ознакомлены с принципами рефакторинга и основными видами рефакторингов. Также у студентов должна быть сформировано представление о взаимном дополнении разработки через тестирование и рефакторинга.

Процесс проектирования, с точки зрения гибких технологий, является фундаментальной фазой процесса разработки программного обеспечения, которая постепенно, проходя через несколько стадий, преобразует системные требования в конечный продукт. Ниже приведены некоторые рекомендации по самостоятельному изучению гибких технологий проектирования и управления требованиями:

- Изучите признаки «плохого проекта» по материалам книги [2, глава 7].
- Изучите стратегию проектирования, применяемую в экстремальном программировании, по материалам книги [1, глава 17].
- Ознакомьтесь с основными принципами гибкого управления требованиями по материалам книги [5, глава 11].
- Ознакомьтесь с основными принципами создания и использования метафоры [3, глава 24].

- Ознакомьтесь с основными признаками «простого дизайна» по материалам книги [3, глава 18].
- Ознакомьтесь с основными паттернами проектирования по материалам книги [7].
- Изучите следующие структурные паттерны:
    - Адаптер (Adapter);
    - Декоратор (Decorator) или Оболочка (Wrapper);
    - Заместитель (Proxy) или Суррогат (Surrogate);
    - Информационный эксперт (Information Expert);
    - Компоновщик (Composite);
    - Мост (Bridge), Handle (описатель) или Тело (Body);
    - Низкая связанность (Low Coupling);
    - Приспособленец (Flyweight);
    - Устойчивый к изменениям (Protected Variations);
    - Фасад (Facade).
- Изучите следующие паттерны проектирования поведения классов:
    - Интерпретатор (Interpreter);
    - Итератор (Iterator) или Курсор (Cursor);
    - Команда (Command), Действие (Action) или Транзакция (Транзакция);
    - Наблюдатель (Observer), Опубликовать - подписаться (Publish - Subscribe) или Delegation Event Model;
    - Не разговаривайте с неизвестными (Don't talk to strangers);
    - Посетитель (Visitor);
    - Посредник (Mediator);
    - Состояние (State);
    - Стратегия (Strategy);

- Хранитель (Memento);
- Цепочка обязанностей (Chain of Responsibility);
- Шаблонный метод (Template Method);
- Высокое зацепление (High Cohesion);
- Контроллер (Controller);
- Полиморфизм (Polymorphism);
- Искусственный (Pure Fabrication);
- Перенаправление (Indirection).

- Изучите следующие порождающие паттерны:
  - Абстрактная фабрика (Abstract Factory, Factory), др. название Инструментарий (Kit);
  - Одиночка (Singleton);
  - Прототип (Prototype);
  - Создатель экземпляров класса (Creator);
  - Строитель (Builder);
  - (Фабричный метод) Factory Method или Виртуальный конструктор (Virtual Constructor).

- Ознакомьтесь с основами гибкого моделирования по материалам книги [9].

Одной из ключевых практик гибких технологий разработки программного обеспечения является также рефакторинг исходного кода. Рефакторингом называется изменение исходного кода программы без изменения его функциональности. Рефакторинг позволяет непрерывно улучшать архитектуру приложения. Принципиально важным является совместное использование рефакторинга и автоматических модульных тестов, так как последние позволяют убедится, что выполненный рефакторинг не нарушил функциональность системы.

- Изучите технологию рефакторинга по материалам книги [8].

- Ознакомьтесь с каталогом рефакторингов по материалам сайта [53].

Обратите внимание на следующие методы рефакторинга:

- Добавление параметра (Add Parameter);
- Замена двусторонней связи односторонней (Change Bidirectional Association to Unidirectional);
- Замена ссылки значением (Change Reference to Value);
- Замена односторонней связи двусторонней (Change Unidirectional Association to Bidirectional);
- Замена значения ссылкой (Change Value to Reference);
- Сворачивание иерархии (Collapse Hierarchy);
- Консолидация условного выражения (Consolidate Conditional Expression);
- Преобразование динамического конструирования в статическое (Convert Dynamic to Static Construction) послал Gerard M. Davison;
- Преобразование статического конструирования в динамическое (Convert Static to Dynamic Construction) послал Gerard M. Davison;
- Декомпозиция условного оператора (Decompose Conditional);
- Дублирование видимых данных (Duplicate Observed Data);
- Устранение внутриклассового сообщения бинов (Eliminate Inter-Entity Bean Communication);
- Инкапсуляция коллекции (Encapsulate Collection);
- Инкапсуляция нисходящего преобразования типа (Encapsulate Downcast);
- Инкапсуляция поля (Encapsulate Field);
- Выделение класса (Extract Class);
- Выделение интерфейса (Extract Interface);
- Выделение метода (Extract Method);
- Выделение пакета (Extract Package);

- Выделение подкласса (Extract Subclass);
- Выделение родительского класса (Extract Superclass);
- Формирование шаблона метода (Form Template Method);
- Скрытие делегирования (Hide Delegate);
- Скрытие метода (Hide Method);
- Встраивание класса (Inline Class);
- Встраивание метода (Inline Method);
- Встраивание временной переменной (Inline Temp);
- Введение утверждения (Introduce Assertion);
- Введение поясняющей переменной (Introduce Explaining Variable);
- Введение внешнего метода (Introduce Foreign Method);
- Введение локального расширения (Introduce Local Extension);
- Введение объекта Null (Introduce Null Object);
- Введение объекта параметров (Introduce Parameter Object);
- Перемещнение класса (Move Class);
- Перемещнение поля (Move Field);
- Перемещение метода (Move Method).

- Изучите противоположные методы рефакторинга, такие как выделение/встраивание метода.
- Ознакомьтесь со средствами автоматического рефакторинга по материалам сайта [54].

Для закрепления пройденного материала, выполните следующие задания:

- Воспользуйтесь технологией рефакторинга для итеративного улучшения качества кода при выполнении лабораторных заданий и курсовых проектов. Проанализируйте ошибки, внесенные в программу во время рефакторинга. Выясните, что являлось причиной ошибок (например, выполнение более чем одного

рефакторинга за один раз, выполнение неправильного рефакторинга, отсутствие модульных тестов на соответствующую функциональность).

- Выберите подходящую систему автоматического рефакторинга и воспользуйтесь ей при выполнении лабораторных заданий и курсовых проектов. Проанализируйте достоинства и недостатки автоматических средств рефакторинга по сравнению с ручным выполнением рефакторинга.

По итогам самостоятельного изучения технологий проектирования и управления требованиями постарайтесь ответить на следующие вопросы:

- Приведите основные признаки «плохого проекта».
- Приведите основные характеристики простого дизайна.
- Обоснуйте целесообразность использования наиболее простого дизайна.
- Обоснуйте целесообразность использования CRC-карточек при проектировании.

Датчик температуры	
Обязанности **– Измерить температуру** **– Подать сигнал тревоги, если значение температуры превышает заданный уровень**	**Взаимодействия** – ***Ремонт*** – ***Резерв***

Рис. 5. Пример CRC-карточки

- Обоснуйте целесообразность специфицирования поведения системы.
- Обоснуйте целесообразность использования технологических прототипов.
- Что такое паттерн проектирования? Приведите примеры.
- В чем состоит отличие рефакторинга от реинжиниринга?

- Приведите основные критерии, на основе которых принимается решение о необходимости рефакторинга.
- Приведите примеры рефакторингов.
- Приведите примеры противоположных рефакторингов и обоснуйте необходимость их применения (в «прямом» и «обратном» случае).
- Обоснуйте целесообразность использования средств автоматического рефакторинга.

Тема семинара: «Гибкое проектирование и моделирование»

Темы докладов:

1. Обзор паттернов проектирования.
2. Обзор технологии рефакторинга.
3. Обзор гибкого моделирования.

5. Планирование и управление проектом

Цель и задачи темы

Сформировать исчерпывающее представление об особенностях управления гибким проектом. Описать особенности гибкого планирования и его отличие от традиционных методик планирования. Сформировать исчерпывающее представление об этапах и практиках гибкого планирования.

Требования к уровню освоения темы

Студенты должны получить исчерпывающее представление об особенностях управления гибким проектом. Студенты должны понимать важность планирования в гибких технологиях разработки программного обеспечения. Студенты должны овладеть основными гибкими практиками планирования и управления рисками.

Процесс планирования является фундаментальной практикой гибких технологий, позволяющей предсказать процесс развития проекта в целом. Планирование процессов разработки программного обеспечения сталкивается с множеством трудностей, связанных со сложностью определения этих процессов и их непредсказуемостью. Гибкие технологии разработки программного обеспечения предлагают методы планирования, которые в большой степени учитывают неопределимость и непредсказуемость процессов разработки программного обеспечения.

Ниже приведены некоторые рекомендации по самостоятельному изучению традиционных технологий планирования:

- Ознакомьтесь с технологией использования диаграмм Ганта по материалам [12, глава 8].
- Ознакомьтесь с технологией PERT (*Program Evaluation and Review Technique*) по материалам сайта [12, глава 8].
- Ознакомьтесь с теорией ограничений и технологией CCPM (Critical Chain Project Management) по материалам сайтов [55].
- Ознакомьтесь с обзором инструментальных средств планирования проекта по материалам сайта [56].
- Ознакомьтесь с критикой традиционный технологий планирования по материалам статьи [57].

Ниже приведены некоторые рекомендации по самостоятельному изучению гибких технологий планирования и управления проектом:

Для

- Ознакомьтесь с основными переменными проекта и взаимосвязях между ними по материалам книги [1, главы 4, 6, 7].

- Изучите стратегию планирования, применяемую в экстремальном программировании, по материалам книги [1, глава 15].

- Изучите основы планирования версий и итераций по материалам книги [5, главы 10, 16-18].

- Изучите методы оценки трудозатрат по материалам книги [5, глава 12].

- Ознакомьтесь с технологией управление личным временем GTD (*Getting Things Done*) по материалам сайта [58].

Управление персоналом оказывает, возможно, наибольшее влияние на успех проекта в целом. Политика в области найма сотрудников и руководства командой во многом определяет успех проекта и успех организации в целом. В области гибких технологий разработки программного обеспечения сложился набор практик, позволяющих эффективно управлять сотрудниками, занятыми в проектах по разработке программного обеспечения. Ниже приведены некоторые рекомендации по самостоятельному изучению методов управления персоналом:

- Изучите стратегию менеджмента по материалам книги [1, глава 12].

- Изучите стратегию организации рабочего места по материалам книги [1, глава 13].

- Ознакомьтесь с основными рекомендациями по управлению персоналом по материалам книги [10].

- Ознакомьтесь с рекомендациями по интервьюированию кандидатов по материалам статьи [59].

Для закрепления пройденного материала, выполните следующие задания:

- Воспользуйтесь технологией гибкого планирования при выполнении лабораторных заданий и курсовых проектов. Проанализируйте результаты использования гибкого планирования по завершению проекта.

По итогам самостоятельного изучения технологий планирования и управления проектом постарайтесь ответить на следующие вопросы:

- Опишите четыре переменных управления проектом: стоимость, качество, сроки и объем работ.

- Опишите основные этапы планирования версий.

- Опишите основные принципы деления проекта на итерации.

- В чем состоит целесообразность организации ежедневных совещаний?
- В чем состоит отличие планирования версий от планирования итераций?
- Перечислите основные метрики проекта?
- Опишите основные приемы гибкого менеджмента проектов.
- Обоснуйте целесообразность накопления человеческого капитала.
- Обоснуйте целесообразность формирования продуктивных команд.
- Приведите примеры способов мотивации, отрицательно влияющих на моральное состояние команды.
- Приведите примеры адекватных способов мотивации сотрудников.
- Приведите общие принципы интервьюирования кандидатов.

Тема семинара: «Планирование и управление проектом»

Темы докладов:

1. Обзор традиционных методов планирования проекта.
2. Обзор адаптивных технологий планирования проекта.
3. Обзор методов мотивации и управления командой разработчиков.

6. Обзор гибких методологий разработки программного обеспечения

Цель и задачи темы

Сформировать общее представление об основных современных гибких методологиях разработки программного обеспечения.

Требования к уровню освоения темы

Студенты должны получить общее представление об основных современных гибких методологиях разработки программного обеспечения.

В настоящее время известны разнообразные гибкие методологии разработки программного обеспечения. Большинство гибких методологий минимизируют риски, возникающие при разработке программного обеспечения разбивая весь процесс разработки продукта на небольшие периоды времени, называемые *итерациями*. Также большинство гибких методологий приветствуют активное и открытое общение между членами команды. Кроме этого, основным критерием успешности проекта рассматривается работающее программное обеспечение.

Ниже приведены некоторые рекомендации по самостоятельному ознакомлению с современными гибкими методологиями разработки программного обеспечения:

- Ознакомьтесь с основными принципами экстремального программирования [1].

- Ознакомьтесь с основными ценностями экстремального программирования [1, глава 7]. Экстремальное программирование основано на следующих ключевых ценностях:

  Коммуникация. Дисциплина XP нацелена на обеспечение непрерывной, постоянно осуществляемой коммуникации между участниками проекта.

  Простота. Дисциплина XP предлагает решать задачу самым простым способом из всех возможных.

  Обратная связь. Дисциплина XP предлагает сделать все возможное, чтобы начать получать отзывы о разрабатываемом продукте как можно раньше и в дальнейшем получать их как можно чаще.

  Храбрость. Дисциплина XP призывает совершать правильные поступки в нужные моменты времени.

- Изучите основные практики экстремального программирования [1, глава 10]. Дисциплина XP содержит следующие практики, которые позволяют непротиворечиво следовать ключевым ценностям:

  - **Игра в планирование** – диалог между заказчиками и разработчиками с целью определить текущие задачи, приоритеты и сроки.

  - **Разработка через тестирование** – частью системы являются автоматические тесты, которые позволяют в любой момент убедиться в корректности работы системы.

  - **Парное программирование** – любой код разрабатывается двумя программистами.

  - **Рефакторинг** – непрерывное улучшение кода путем его переработки.

  - **Простой дизайн** – постоянное видоизменение дизайна системы с целью его упрощения и очищения.

  - **Коллективное владение кодом** – все члены команды обладают правом вносить изменения в любой код системы.

  - **Постоянная интеграция** – интеграция продукта выполняется постоянно, по несколько раз в день.

  - **Заказчик на месте разработки** – рядом с разработчиками постоянно находится представитель заказчика, который работает вместе с ними.

  - **Частые выпуски версий** – новые версии продукта выпускаются и внедряются в эксплуатацию как можно чаще.

  - **40-часовая рабочая неделя** – разработчики не должны работать сверхурочно, так как от этого снижается их производительность и качество их работы.

  - **Стандарты кодирования** – все участники команды должны придерживаться общих стандартов кодирования.

  - **Метафора системы** – простая аналогия, интуитивно понятная всем участникам проекта, которая в доступной форме описывает строение и функционирование системы.

- Изучите взаимодействие основных практик экстремального программирования [1, глава 11].

- Ознакомьтесь с особенностями внедрения экстремального программирования [1, часть 3] .

- Ознакомьтесь с основными принципами методологии Scrum [57]. Методология *Scrum* устанавливает простые и понятные правила управления процессом разработки и позволяет использовать уже существующие практики проектирования и кодирования, итеративно корректируя требования к продукту.

- Ознакомьтесь с принципами разделения ролей в методологии Scrum [57]. В методологии *Scrum* формально выделены три роли, для каждой из которых формально определены зоны прав и ответственности: *владелец продукта* (*Product Owner*), *команда* (*Team*) и *скрам-мастер* (*ScrumMaster*).

- Ознакомьтесь с основными фазами проекта в методологии Scrum [57]. Основой *Scrum* является итеративная разработка. *Scrum* определяет итеративные правила управления проектом, которые призваны обеспечивать достижение максимального эффекта от реализованной функциональности.

- Ознакомьтесь с основными видами документации методологии Scrum [57]. Методология *Scrum* требует наличия всего трех типов формальных документов: журнала продукта, журнала спринта и графика спринта.

- Ознакомьтесь с основными принципами методологии Dynamic Systems Development Method [60]. Методология DSDM позиционируется как наиболее зрелая методология гибкой разработки программного обеспечения. Многие другие гибкие методологии сфокусированы на *программировании*, в то время как методология DSDM сфокусирована на организации процесса производства программного обеспечения.

- Ознакомьтесь с основными фазами проекта в методологии Dynamic Systems Development Method [60].

- Ознакомьтесь с основными стадиями жизненного цикла проекта в методологии Dynamic Systems Development Method [60]:

    1. изучение выполнимости проекта (*The Feasibility Study*);

2. изучение бизнес-процессов (*The Business Study*);
3. итерация разработки функциональной модели (*Functional Model Iteration*);
4. итерация разработки (*Design and Build Iteration*);
5. внедрение (*Implementation*).

- Ознакомьтесь с основными практиками методологии Dynamic Systems Development Method [60]:
    1. ограничение по времени (*Timeboxing*);
    2. принцип назначения приоритетов *MoSCoW*;
    3. прототипирование;
    4. тестирование;
    5. обсуждение в режиме семинара (*Workshop*);
    6. моделирование;
    7. управление конфигурацией.
- Ознакомьтесь с основными ролями методологии Dynamic Systems Development Method [60]:
    1. чемпион проекта (*Project Champion* или *Executive Sponsor*);
    2. провидец (*Visionary*);
    3. представительный пользователь (*Ambassador User*);
    4. пользователь-консультант (*Advisor User*);
    5. менеджер проект (*Project Manager*);
    6. технический координатор (*Technical Co-ordinator*);
    7. лидер команды (*Team Leader*);
    8. разработчик (*Developer*);
    9. тестировщик (*Tester*);
    10. секретарь (Scribe);

11. посредник (*Facilitator*).

- Ознакомьтесь с основными факторами успеха проекта в методологии Dynamic Systems Development Method [60].

- Ознакомьтесь с основными принципами методологии Feature Driven Development [61]. В отличие от других гибких методологий разработки программного обеспечения, таких как экстремальное программирование, методология *FDD* обладает большим формализмом, но при этом позволяет полнее контролировать процесс разработки.

- Изучите пять основных фаз проекта в методологии Feature Driven Development [61]:

    1. разработка общей модели (*Develop Overall Model*);
    2. построение списка необходимой функциональности (*Build Feature List*);
    3. планирование на основе списка функциональности (*Plan By Feature*);
    4. проектирование функциональности (*Design By Feature*);
    5. реализация функциональности (*Build By Feature*).

- Изучите шесть основных контрольных точек методологии Feature Driven Development [61]:

    1. анализ прикладной области (*Domain Walkthrough*);
    2. проектирование (*Design*);
    3. проверка дизайна (*Design Inspection*);
    4. кодирование (*Code*);
    5. проверка кода (*Code Inspection*);
    6. интеграция (*Promote To Build*).

- Изучите основные практики методологии Feature Driven Development [61]:

    1. моделирование предметной области (*Domain Object Modeling*);

2. функционально-ориентированная разработка (*Developing by Feature*);
3. индивидуальное владение кодом (*Individual Code Ownership*);
4. функционально-ориентированные команды (*Feature Teams*);
5. инспектирование (*Inspections*);
6. управление конфигурацией (*Configuration Management*);
7. непрерывная интеграция (*Regular Builds*);
8. прозрачность результатов (*Visibility of progress and results*).

По итогам самостоятельного изучения технологий планирования и управления проектом постарайтесь ответить на следующие вопросы:

- Опишите основные принципы экстремального программирования.
- Опишите основные ценности экстремального программирования.
- Опишите состав основных практик экстремального программирования.
- Опишите взаимодействие и взаимодополнение основных практик экстремального программирования.
- Опишите особенности внедрения экстремального программирования.
- Опишите основные принципы методологии Scrum.
- Опишите принципы разделения ролей в методологии Scrum.
- Опишите основные фазы проекта в соответствии с методологией Scrum.
- Опишите основные виды документов, которые используются для отслеживания процессов в методологии Scrum.
- Опишите основные принципы методологии Dynamic Systems Development Method.

- Опишите основные фазы проекта в методологии Dynamic Systems Development Method.
- Опишите основные стадии жизненного цикла проекта методологии Dynamic Systems Development Method.
- Опишите основные практики методологии Dynamic Systems Development Method.
- Опишите принцип назначения приоритетов MoSCoW в методологии Dynamic Systems Development Method.
- Опишите основные роли в методологии Dynamic Systems Development Method.
- Опишите основные принципы методологии Feature Driven Development.
- Опишите основные фазы проекта в методологии Feature Driven Development.
- Опишите основные контрольные точки проекта в методологии Feature Driven Development.
- Опишите основные практики методологии Feature Driven Development.

Тема семинара: «Обзор гибких методологий разработки программного обеспечения»

Темы докладов:

1. Обзор методологии XP.
2. Обзор методологии Scrum.
3. Обзор методологии Dynamic Systems Development Method.
4. Обзор методологии Feature Driven Development.

Основная литература по модулю

1. Бек К. Экстремальное программирование. – Спб.: Питер, 2002.
2. Мартин Р., Ньюкирк Д., Косс Р. Быстрая разработка программ. Принципы, примеры, практика. – М.: Издательский дом “Вильямс”, 2004.
3. Ауер К., Миллер Р. Экстремальное программирование: постановка процесса. С первых шагов и до победного конца. – Спб.: Питер, 2004.
4. Бек К. Экстремальное программирование: разработка через тестирование. – Спб.: Питер, 2003.
5. Бек К. Фаулер М. Экстремальное программирование: планирование. – Спб.: Питер, 2003.
6. Астелс Д., Миллер Г., Новак М. Практическое руководство по экстремальному программированию. – М.: Издательский дом “Вильямс”, 2002.

Дополнительная литература

7. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – СПб.: Питер, 2001.
8. Рефакторинг: улучшение существующего кода / М. Фаулер, К. Бек, Д. Брант и др. – СПб.: Символ-Плюс, 2004.
9. Амблер С. Гибкие технологии: экстремальное программирование и унифицированный процесс разработки. – Спб.: Питер, 2004.
10. Демарко Т., Листер Т. Человеческий фактор: успешные проекты и команды. – М.: Символ-Плюс, 2005.
11. Макконнелл С. Совершенный код. – СПб.: Питер, 2007.
12. Гецци К., Мехди Дж., Мандриоли Д. Основы инженерии программного обеспечения. 2-е издание. – Санкт-Петербург: БХВ-Петербург, 2005.
13. Крачтен Ф. Введение в Rational Unified Process. 2-е издание. – М.: Издательский дом "Вильямс", 2002.
14. Паук М., Куртис Б., Хриссис М. Б. Модель зрелости процессов разработки программного обеспечения. – М.: Интерфейс-Пресс, 2002.

Дополнительные статьи и Интернет-ресурсы:

15. Бах Дж. Незрелость CMM (The Immaturity of CMM), http://www.bcc.ru/press/articles/immaturity_of_cmm.html

16. Статья “Agile software development” в Wikipedia, http://en.wikipedia.org/wiki/Agile_software_development

17. Официальный сайт проекта jUnit, http://www.junit.org/

18. Официальный сайт проекта NUnit, http://www.nunit.org/

19. Реестр фреймворков тестирования для различных языков программирования, http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

20. Официальный сайт проекта TestDriven.NET, http://www.testdriven.net/

21. Prohorenko A., Prohorenko O. Using JUnit With Eclipse IDE, http://www.onjava.com/pub/a/onjava/2004/02/04/juie.html

22. Официальный сайт проекта RSpec, http://rspec.rubyforge.org/

23. Введение во фреймворк RSpec http://blog.davidchelimsky.net/articles/2007/05/14/an-introduction-to-rspec-part-i

24. Fowler M. Mocks Aren't Stubs, http://martinfowler.com/articles/mocksArentStubs.html

25. Mackinnon T., Freeman S., Craig P. Endo-Testing: Unit Testing with Mock Objects, http://mockobjects.com/files/endotesting.pdf

26. Freeman S., Mackinnon T., Pryce N., Walnes J. Mock Roles, Not Objects, http://mockobjects.com/files/mockrolesnotobjects.pdf

27. Статья “Mock object” в Wikipedia, http://en.wikipedia.org/wiki/Mock_object

28. Официальный сайт проекта EasyMock, http://www.easymock.org/

29. Официальный сайт проекта Rhino.Mocks, http://www.ayende.com/projects/rhino-mocks.aspx

30. Официальный сайт проекта NUnitForms, http://nunitforms.sourceforge.net/

31. Официальный сайт проекта jfcUnit, http://jfcunit.sourceforge.net/

32. Официальный сайт проекта DbUnit, http://dbunit.sourceforge.net/

33. Официальный сайт проекта HttpUnit, http://httpunit.sourceforge.net/

34. Статья “Naming conventions (programming)” в Wikipedia, http://en.wikipedia.org/wiki/Naming_conventions_(programming)

35. Статья “Programming style” в Wikipedia, http://en.wikipedia.org/wiki/Coding_standard

36. Code Conventions for the Java Programming Language, http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

37. Coding Standard: C#, http://www.tiobe.com/standards/gemrcsharpcs.pdf

38. Stephens M. The perils of pair programming, http://www.regdeveloper.co.uk/2007/01/31/perils_pair_programming/

39. Collins-Sussman B., Fitzpatrick B., Pilato M. Version Control with Subversion, O'Reilly, 2004, http://svnbook.red-bean.com/en/1.2/index.html

40. Статья “Comparison of revision control software” в Wikipedia, http://en.wikipedia.org/wiki/Comparison_of_revision_control_software

41. Официальный сайт продукта Visual SourceSafe, http://msdn.microsoft.com/ssafe/

42. Официальный сайт проекта Apache Ant, http://ant.apache.org/

43. Официальный сайт проекта NAnt, http://nant.sourceforge.net/

44. Официальный сайт продукта Visual Studio MSBuild, http://msdn2.microsoft.com/en-us/library/wea2sca5(VS.80).aspx

45. Lee K. Architecting the Build Process, http://www.buildmeister.com//modules/content/index.php?id=40

46. Cymerman M. Automate your build process using Java and Ant, http://www.javaworld.com/javaworld/jw-10-2000/jw-1020-ant.html?page=1

47. Официальный сайт проекта CruiseControl, http://cruisecontrol.sourceforge.net/

48. Официальный сайт проекта CruiseControl.NET, http://ccnet.thoughtworks.com/

49. Grenyer P. Continuous Integration with CruiseControl.Net, http://www.marauder-consulting.co.uk/Continuous_Integration_with_CruiseControl_Net.pdf

50. Nguyen H. Tracking Down a Defect Management Tool, http://www.stickyminds.com/sitewide.asp?Function=edetail&ObjectType=MOVED&ObjectId=2993

51. Статья “Bug tracking system” в Wikipedia, http://en.wikipedia.org/wiki/Bug_tracking_system

52. Статья “Application Lifecycle Management” в Wikipedia, http://en.wikipedia.org/wiki/Application_Lifecycle_Management

53. Каталог рефакторингов, http://www.refactoring.com/catalog/index.html

54. Каталог инструментов для автоматического рефакторинга, http://www.refactoring.com/tools.html

55. Статья “Critical Chain Project Management” в Wikipedia, http://en.wikipedia.org/wiki/Critical_Chain

56. Статья “List of project management software” в Wikipedia, http://en.wikipedia.org/wiki/List_of_project_management_software

57. Schwaber K. SCRUM Development Process, http://jeffsutherland.com/oopsla/schwapub.pdf

58. Статья “Getting Things Done” в Wikipedia, http://en.wikipedia.org/wiki/Getting_Things_Done

59. Спольски Дж. Искусство интервью, http://rsdn.ru/article/career/interv.xml

60. Статья “Dynamic Systems Development Method” в Wikipedia, http://en.wikipedia.org/wiki/DSDM

61. Статья “Feature Driven Development” в Wikipedia, http://en.wikipedia.org/wiki/Feature_Driven_Development