Министерство образования Московской области

Государственное бюджетное профессиональное

образовательное учреждение Московской области

Воскресенский колледж

**Методические указания и контрольные задания**

**по дисциплине «Методы проектирования программного обеспечения»**

для обучающихся по заочной форме

по специальности **09.02.03 Программирование в компьютерных системах**

**Воскресенск, 2015 г.**

Разработчик(и):

ГБПОУ СПО МО «Воскресенский колледж» преподаватель спецдисциплин Комиссаров С.А.

(место работы) (занимаемая должность) (инициалы)

Одобрено на заседании предметно-цикловой комиссии

компьютерных дисциплин

Протокол №\_\_\_\_\_\_\_\_от «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_г.

Председатель ПЦК\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

**Пояснительная записка**

Методические указания для студентов заочного отделения по выполнению практических заданий по дисциплине «Методы проектирования программного обеспечения» предназначены для реализации ФГОС по специальности 09.02.03 «Программирование в компьютерных системах».

«Методы проектирования программного обеспечения» входит в профессиональный модуль ПМ03 «Участие в интеграции программных модулей».

Обучающийся в ходе освоения учебной дисциплины должен:

***Уметь:***

владеть основными методологиями процессов разработки программного обеспечения;

использовать методы для получения кода с заданной функциональностью и степенью качества.

***Знать:***

модели процесса разработки программного обеспечения;

основные принципы процесса разработки программного обеспечения;

основные методы и средства эффективной разработки;

методы организации работы в коллективах разработчиков программного обеспечения;

стандарты качества программного обеспечения.

Каждая контрольная работа разделена на 10 вариантов. Вариант работы определяется по последней цифре индивидуального шифра.

Перед выполнением контрольной работы необходимо изучить теоретический материал по данной теме, который достаточно полно представлен в списке литературы. Список разделов учебной дисциплины для изучения приведен после примера выполнения задания.

В процессе выполнения работы рекомендуется тщательно анализировать имеющиеся в учебниках иллюстрации, схемы, диаграммы и таблицы.

Необходимо обязательно выяснять значение встречающихся в тексте незнакомых или непонятных слов и терминов.

Также при подготовке к выполнению контрольной работы нужно пользоваться источниками, доступными в сети интернет. К ее выполнению следует приступать после того, как вы освоите достаточное количество материала, чтобы выполнить все задания по возможности одновременно.

**Контрольная работа** включает изучение разделов «Общие сведения о программных продуктах», «Технологии разработки программных продуктов» и «Современные технологии программирования».

**Требования к оформлению работы.**

Работа выполняется на листах формата А4 в печатном виде. Шрифт – Times New Roman, размер шрифта – 12, междустрочный интервал – полуторный. Размеры полей: слева – 2 см (под подшивку), остальные – по 1 см (сверху, снизу, справа).

В начале работы, после титульного листа, указывается содержание работы, в конце – список использованной литературы. При оформлении любого задания необходимо сначала указать вопрос.

**Пример правильного ответа**.

**Вопрос: Технологии коллективной разработки программного обеспечения.**

**Ответ.**

Коллективная разработка — это бизнес-стратегия, рабочий процесс и набор программного обеспечения, способствующие совместной работе различных организаций, программистов над одним изделием.

Плодотворная разработка различных программных комплексов на сегодняшний день немыслима без использования сопутствующих современных технологий программирования, функционирование которых направленно на решение проблем, непосредственно связанных с созданием программного обеспечения, обладающего такими качествами, как надёжность, модульность, информативность, способность к видоизменению, повторная используемость кода и тому подобное.

Указанные качества необходимы при создании программного обеспечения для различных областей деятельности, исходя из этого в сфере разработки новых технологий программирования, в том числе для параллельного программирования, ведутся интенсивные научные исследования.

Для того чтобы создать крупную информационную систему необходимо задействовать группу согласованно работающих программистов. Из практики видно, что благодаря появлению систем быстрой разработки приложений, организация взаимосвязанной работы нескольких программистов может потребоваться не только крупному, но и малому бизнесу. Так, к примеру. внедрение средств Rapid Application Development (RAD) даёт возможность увеличить производительность коллективных разработок, путём сокращения численности их групп и затрачиваемого времени. В итоге потребность в обеспечении коллективных разработок малых группам повлекла за собой появление на рынке множество программ.

Коллективная разработка программного обеспечения включает как минимум следующие процессы:

· Разработка

· Тестирование

· Сборка

· Развёртывание

· Выпуск продукта.

Системы коллективной разработки должны выполнять как минимум две функции — это регистрация изменений, вносимых в проект и хранение файлов проекта.

Термин «collaborate» (с англ. коллективная работа) в XXI веке, веке технологий, популярен как никогда. Он означает, что сервис предоставляет возможность одновременной работы сразу для нескольких пользователей, которые трудятся над решением общей задачи.

У одних сервисов коллективный труд является лишь одной из опций, а у других — основной функцией. Основной проблемой является то, что термин “collaborate” не всегда обозначает «командную работу», а среди сервисов, по-настоящему поддерживающих совместную работу, бесплатны лишь некоторые.

Система управления версиями представляет собой программное обеспечение, для работы с постоянно изменяющейся информацией. Она даёт возможность хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним, а также определять, кто и когда сделал то или иное изменение, и многое другое.

Подобные системы довольно широко используются в процессе разработки программного обеспечения для сохранения исходных кодов создаваемой программы, также они могут с успехом применяться и в других областях программирования, при работе с большим количеством постоянно изменяющихся электронных документов.

Рассмотрим некоторые из них.

Bazaar, ранее известная как Bazaar-NG, утилита командной строки bzr, — это распределённая система управления версиями, разработка которой спонсируется фирмой Canonical Ltd, в последнюю версию по сравнению с предыдущей было внесено более 50 изменений. Данная система разработана в целях облегчения создания и развития проектов для пользователей.

Mercurial, в переводе с англ. «подвижный», — распределённая система управления версиями, способная функционировать на многих операционных системах и различных аппаратных платформах, разработанная для эффективной работы с очень большими кодами.

· Git — распределённая система управления версиями файлов. Код программы был написан на языке «С», проект создан Линусом Торвальдсом в 2005 году для управления разработкой ядра Linux, является общедоступным программным обеспечением. Данная система была введена многими ведущими разработчиками, используется в известных Linux-сообществу проектах.

Concurrent Versions System (или CVS, в переводе «Система Одновременных Версий») — представляет собой программный продукт, который относится к разряду систем управления версиями. Программа хранит историю изменений исходного кода программного обеспечения, тем самым облегчая совместную работу программистов над одним проектом. CVS популярна в мире открытого программного обеспечения.

Bazaar — удобная система контроля версий с приятным интерфейсом, она хорошо подходит для пользователей, которых не привлекает перспектива работы с командной строкой. Имеется множество дополнительных опций и расширений, что позволяет настроить программу под свои нужды.

Говоря о Mercurial следует отметить, что простой и отточенный интерфейс, и набор команд, возможность импортировать репозиториев с других систем контроля версий, — сделают переход на данную программу безболезненным и быстрым, а её надёжность и скорость работы позволяют пользоваться им для контроля версий огромных проектов. Все это позволяет Mercurial стать достойным конкурентом git’а.

В свою очередь Git — это гибкая, удобная и мощная система контроля версий, способная удовлетворить абсолютное большинство пользователей. Git — один из лидеров систем контроля версий.

Несмотря на то, что программа CVS достаточно устарела и обладает весомыми недостатками, она все ещё является одной из самых популярных систем контроля версий и отлично подходит для управления малыми проектами, не требующих создания нескольких параллельных версий, которые надо периодически соединять.

Большой выбор систем контроля версий позволяет удовлетворить любые требования и организовать работу так, как вам необходимо. Однако, среди всего многообразия систем есть явный лидер, в итоге проведённого анализа на лидирующее место выдвигается программа Git.

Однако, выбор — это всегда «дело вкуса», системы контроля версий — представляет собой программное обеспечения необходимое для каждого разработчика и не только, они позволяют хранить несколько версий одного и того же документа, и при необходимости возвращаться к более ранним версиям.

На рынке существует множество подобных инструментов, в данной статье рассмотрена лишь их часть, с целью продемонстрировать достоинства и недостатки тех или иных их видов.

Конкуренция на рынке также достаточна высока, в связи с чем компании по выпуску программ для коллективных разработок проектов, с каждом годом выпускают усовершенствованные версии своих «детищ».

В заключение представлена таблица сравнения вышеописанных систем.

**Анализ  программ  по  коллективной  разработке  ПО**

|  |  |  |
| --- | --- | --- |
| **Название** | **Преимущества** | **Недостатки** |
| Bazaar | ·                   не  требует  использования  специального  сервера,  поддерживает  работу  как  с  ним,  так  и  без  него;  ·                   возможность  создавать  новые  ветки  на  основе  репозиториев  других  систем;  ·                   поддерживает  полный  набор  символов  Unicode  в  именах  файлов  ·                   кроссплатформенная  поддержка. | ·                более  низкая  скорость  работы,  по  сравнению  с  Git  и  Mercurial;  ·                необходима  установка  большого  количества  плагинов, |
| Mercurial | ·               кроссплатформенная  поддержка.  ·               возможность  работы  с  несколькими  ветками  проекта.  ·               быстрая  обработка  данных.  ·               проста  в  обращении.  ·               возможность  конвертирования  репозиториев  иных  систем  поддержки  версий,  таких  как  CVS,  Subversion,  Git,  Darcs,  GNU  Arch,  Bazaar  и  др. | ·                возможны  совпадения  хеш-кода  отличных  по  содержанию  ревизий.  ·                Ориентирована  только  на  работу  в  консоли. |
| Git | ·               надёжная  система  сравнения  ревизий  и  проверки  корректности  данных;  ·               эластичная  система  ветвления  проектов  и  слияния  веток  между  друг  другом.  ·               наличие  локального  репозиториев  позволяет  вести  полноценный  локальный  контроль  изменений  ·               высокая  производительность  и  скорость  работы;  ·               удобный  и  интуитивно  понятный  интерфейс;  ·               множество  графических  оболочек;  ·               возможность  делать  контрольные  точки,  в  которых  данные  сохраняются  полностью;  ·               широкая  распространённость,  лёгкая  доступность  и  качественная  документация.  ·               гибкость  системы  позволяет  удобно  её  настраивать  и  создавать  специализированные  контроль-системы  или  пользовательские  интерфейсы  на  базе  Git.  ·               универсальный  сетевой  доступ  с  использованием  протоколов  http,  ftp,  rsync,  ssh  и  др. | ·                отсутствует  зрелая  реализация  Git,  совместимая  с  иными  операционными  системами;  ·                совпадения  хеш-кода  отличных  по  содержанию  ревизий;  ·                не  отслеживается  изменение  отдельных  файлов,  а  только  всего  проекта  целиком;  ·                требуется  достаточно  длительное  время  для  скачивания  данных,  особенно,  если  проект  большой. |
| CVS | ·               несколько  клиентов  могут  одновременно  работать  над  одним  и  тем  же  проектом.  ·               позволяет  управлять  не  одним  файлом,  а  целыми  проектами.  ·               обладает  большим  количеством  удобных  графических  интерфейсов,  способных  удовлетворить  практически  любой,  даже  самый  требовательный  вкус.  ·               широко  распространена  и  поставляется  по  умолчанию  с  большинством  операционных  систем  Linux.  ·               при  загрузке  тестовых  файлов  из  репозиториев  передаются  только  изменения,  а  не  весь  файл  целиком. | ·           при  перемещении  или  переименовании  файла,  директории,  теряются  все  привязанные  изменения.  ·           сложности  при  ведении  нескольких  параллельных  веток  одного  и  того  же  проекта.  ·           ограниченная  поддержка  шрифтов.  ·           для  каждого  изменения  бинарного  файла  сохраняется  вся  версия  файла,  а  не  только  внесённые  изменение.  ·           с  клиента  на  сервер  изменённый  файл  всегда  передаётся  полностью.  ·           ресурсоёмкие  операции,  так  как  требуют  частого  обращения  к  репозиториев,  и  сохраняемые  копии  имеют  некоторую  избыточность. |

**ВАРИАНТ №1**

1. Жизненный цикл программного обеспечения. *В ответ включить определение жизненного цикла, самые распространенные модели ЖЦ и описание ключевых этапов ЖЦ*.
2. Системное программное обеспечение: базовое и сервисное. *В ответ включить определение, классификации системного ПО, подробное описание двух-трех программ системного назначения.*
3. Облачные технологии проектирования программного обеспечения. *В ответ включить описание сути облачных технологий разработки, преимущества и недостатки, тенденции развития и самые популярные сервисы.*

**ВАРИАНТ №2**

1. Основы разработки мультиплатформенных приложений. *Дать определение таких приложений, привести примеры, перечислить особенности таких приложений и принципиальные отличия от разработки ПО для одной среды*.
2. Механизмы обратной связи с потребителем в процессе эксплуатации программного обеспечения. *В ответе необходимо дать обоснование необходимости наличия обратной связи между разработчиком и потребителем и подробно описать эти методы.*
3. Стандартные технологические процессы ISO при разработке программного обеспечения. *Пояснить, зачем нужны стандарты при разработке ПО, перечислить и кратко описать технологические процессы.*

**ВАРИАНТ №3**

1. Качество программного обеспечения и стандартизация. *В ответе перечислить критерии качества ПО, кратко их описать, обосновать необходимость стандартов качества ПО и назвать самые распространённые из них.*
2. Классические технологические процессы при разработке программного обеспечения: специфицирование и планирование. *В ответе необходимо подробно описать суть этих двух процессов и их место в ходе разработки ПО.*
3. Языки программирования. *В ответе дать определение ЯП и перечислить классификации ЯП с примерами.*

**ВАРИАНТ №4**

1. Лицензирование и защита программного обеспечения. *В ответе подробно рассказать о необходимости лицензирования и защиты, дать определения этих понятий*.
2. Классификация систем программирования. *Дать определение и развернутую классификацию СП.*
3. Характеристики программного продукта. *Перечислить характеристики ПП, дать их краткое описание.*

**ВАРИАНТ №5**

1. Пакеты прикладных программ. *В ответ включить определение ППП, их классификацию, привести примеры наиболее популярных пакетов.*
2. Классические технологические процессы при разработке программного обеспечения: определение дизайна и визуализация. *В ответе необходимо подробно описать суть этих двух процессов и их место в ходе разработки ПО.*
3. Модульное программирование. *В ответе подробно описать назначение, преимущества и сущность данного типа программирования.*

**ВАРИАНТ №6**

1. Разработка программного продукта: использование технологий OLE и COM. *В ответе необходимо подробно описать суть этих технологий и спектр их применения в современных ПП*.
2. Использование современных интернет-технологий при продаже, распространении и лицензировании программного обеспечения. *В ответе пояснить сущность этих технологий, преимущества и недостатки по сравнению с традиционными способами работы с клиентами.*
3. Классификация программных продуктов. *Требуется развернутый ответ с перечислением возможных классификаций ПП по разным признакам*. *Также в ответе привести примеры наиболее популярных программ в каждом классе.*

**ВАРИАНТ №7**

1. Спиральная модель жизненного цикла программного продукта. Сравнение с другими моделями жизненного цикла. *В ответ включить подробное описание этапов данного типа цикла, указать его достоинства и недостатки, сделать сравнительную таблицу различных моделей ЖЦ.*
2. Классические технологические процессы при разработке программного обеспечения: тестирование и отладка. *В ответе необходимо подробно описать суть этих двух процессов и их место в ходе разработки ПО.*
3. Современные тенденции слияния программного обеспечения для персональных компьютеров и мобильных устройств. *В ответе необходимо привести результаты краткого анализа тенденций развития современного ПО, указать основные направления развития, плюсы и минусы мультиплатформенности.*

**ВАРИАНТ №8**

1. Классические технологические процессы при разработке программного обеспечения: ввод программы в действие, сопровождение, эксплуатация. *В ответе необходимо подробно описать суть этих двух процессов и их место в ходе разработки ПО*.
2. Определение инструментария программирования. *В ответе дать подробное обоснование выбора языка и среды программирования в зависимости от текущих задач. Проиллюстрировать свой ответ конкретным примером.*
3. Основные технологические подходы к разработке программного обеспечения и их классификация. *Требуется развернутый ответ с перечислением основных подходов и способов их классификации*.

**ВАРИАНТ №9**

1. Распределение ролей при разработке программных продуктов. *В ответ включить описание принципа определения ролей при разработке, указать должности и должностные обязанности сотрудников-разработчиков. Проиллюстрировать на конкретном примере.*
2. Порядок разработки тестов программного обеспечения. Автоматизация тестирования. *В ответ включить краткое описание самой сущности тестирования ПО, описать последовательность тестов и обосновать их необходимость, дать определение тесовых скриптов. Проиллюстрировать на конкретном примере.*
3. Разработка программных продуктов: создание системы защиты. *В ответе необходимо подробно описать суть этого процесса и его место в ходе разработки ПО.*

**ВАРИАНТ №10**

1. Системы программирования. Основные понятия и определения. *В ответе дать определение СП и сопутствующих терминов и перечислить классификации СП с примерами.*
2. Разработка программных продуктов: анализ и проектирование с учетом требований стандартов. *В ответе необходимо подробно описать суть этих двух процессов и их место в ходе разработки ПО.*
3. Подход к разработке программных продуктов: опыт крупнейших зарубежных и российских компаний. *В ответе необходимо подробно рассмотреть подход (доктрины) двух-трёх известных компаний к стратегии развития и методикам создания программного обеспечения*.

**Литература:**

1. Гагарина Л.Г., Виснадул Б.Д., Игошин А.В. Ос-новы технологии разработки программных продуктов: Учеб.пособие. – М.: ФОРУМ: ИНФРА-М, 2006. – 192 с.

2. Плаксин М.А. Тестирование и отладка программ – для профессионалов будущих и настоящих. – М,: БИНОМ. Лаборатория знаний, 2007. – 167 с., ил.

3. Microsoft Corporation. Принципы проектирования и разработки программного обеспечения. Учебный курс MCSD/Пер. с англ. – 2-е изд., испр. – М.: Издательско-торговый дом «Русская редакция», 2002. – 736 стр.: ил.

Интернет-источники:

http://www.protesting.ru/

http://cde.osu.ru/

http://info-tehnologii.ru/

http://bourabai.ru/

http://www.intuit.ru/