

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное  
учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем  
в управлении и проектировании (КСУП)

СИСТЕМЫ ВЕРСИОННОГО КОНТРОЛЯ: СИСТЕМА КОНТРОЛЯ  
ВЕРСИЙ MERCURIAL

Пояснительная записка к курсовому проекту по дисциплине  
«Лингвистическое и программное обеспечение САПР» (ЛиПОСАПР)

Студент гр. 583-1

\_\_\_\_\_ Болдырев Е.В.

« \_\_\_\_ » \_\_\_\_\_ 2016 г.

Руководитель ст.

преподаватель каф. КСУП

\_\_\_\_\_ Борисов С.И.

« \_\_\_\_ » \_\_\_\_\_ 2016 г.

Томск 2016

## **Реферат**

Курсовая работа 27 с., 24 рис., 2 таб., 15 источников.

Целью данной работы является обзор существующих систем контроля версий и сравнение этих систем с инструментом Mercurial.

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное  
учреждение высшего образования

ТОСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем  
в управлении и проектировании (КСУП)

Утверждаю

Зав. кафедрой КСУП

\_\_\_\_\_ Ю.А. Шурыгин

«\_\_\_» \_\_\_\_\_ 2016 г.

### ЗАДАНИЕ

на курсовой проект по дисциплине «ЛиПОСАПР»

Студенту гр. 583-1 Болдыреву Евгению Владимировичу

Тема: системы версионного контроля: система контроля версий  
Mercurial

Пояснительная записка к курсовому проекту включает в себя  
следующие разделы:

1) Введение

- Описание предметной области
- Обзор существующих инструментов

2) <Инструмент>

- Краткая характеристика
- Установка серверной части
- Установка клиентской части
- Возможности по интеграции

- Основные сценарии использования
  - Возможность интеграции в IDE Visual Studio
- 3) Сравнение с другими системами
- Сводная сравнительная таблица
  - Выводы из сравнения
- 4) Заключение
- 5) Список литературы

Дата выдачи задания: «1» ноября 2016 г.

Срок сдачи студентом проекта: \_\_\_\_\_.2016

Руководитель ст.

преподаватель каф. КСУП

Борисов С.И. \_\_\_\_\_

Задание принял к исполнению

студент гр. 583-1

Болдырев Е.В. \_\_\_\_\_

## Содержание

1 Введение.....	6
1.1 Описание предметной области .....	6
1.2 Обзор существующих инструментов.....	7
1.2.1 Mercurial.....	7
1.2.2 Git.....	8
1.2.3 Concurrent Version System .....	9
1.2.4 Subversion.....	9
1.2.5 Perforce .....	10
2 Mercurial .....	11
2.1 Краткая характеристика .....	11
2.2 Инсталляция клиентской части .....	12
2.3 Основные сценарии использования .....	13
2.3.1 Сценарии для command-line.....	13
2.3.2 Сценарии для GUI.....	15
2.4 Возможности интеграции IDE Visual Studio.....	24
3 Сравнение с другими системами.....	27
3.1 Сводная сравнительная таблица.....	27
3.2 Выводы из сравнения .....	28
4 Заключение .....	29
Список использованных источников .....	30

# 1 Введение

## 1.1 Описание предметной области

На данный момент в мире существует огромное количество сложных систем, и есть необходимость изменять электронные документы на различных стадиях разработки. Со времени создания документа до окончания проекта, он может потерпеть большое количество изменений, но бывает необходимость использовать промежуточных вариантов документа, а не только последнюю версию. Конечно можно хранить несколько вариантов документов, придавать именам файлов даты их изменений и т.д., но данный способ не эффективен: тратиться большой объем памяти так как приходится хранить почти идентичные файлы, название документов слишком длинное, большая вероятность ошибки, проблематично соблюдать все правила изменений (названий: к примеру “General\_data\_15\_07\_16\_time\_13\_45.cs” и т.д.), так как документ за короткий промежуток времени может измениться большое количество раз, так же замедляется время работы в результате поиска необходимых файлов.

Что бы решить данные проблемы были разработаны программные средства, которые упрощают данный механизм. Данные средства именуются системами контроля версий. Существует несколько систем, каждая из которых является актуальной при определенных условиях их использования.

## 1.2 Обзор существующих инструментов

Система управления версиями (от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое [1].

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов [1].

Список рассматриваемых систем в курсовой работе:

- Mercurial;
- Git (Кузнецов С.В. Система контроля версий Git по сравнению с другими инструментами. Томск, 2016);
- Concurrent Version System (Скрябина Т.С. Система контроля версий CVS по сравнению с другими инструментами. Томск, 2016);
- Subversion (Харченко И.К. Система контроля версий Subversion по сравнению с другими инструментами. Томск, 2016);
- Perforce (Чухнов Д.В. Система контроля версий Perforce по сравнению с другими инструментами. Томск, 2016).

### 1.2.1 Mercurial

Полное название: Mercurial

Краткое название: Hg

Разработчик: Matt Mackall

Лицензия: GNU GPL 2

Первый выпуск: 19 апреля 2005

Последняя версия: 4.0.0 (1 ноября 2016)

Платформа для серверной части: LShift

Платформа для клиентской части: TortoiseHg, gquilt, gwsnhg, HgWin, SourceTree

Платформа: Windows, MacOS, CentOS, Fedora, Debian, Ubuntu, Mandriva, openSUSE, Ark Linux, Arch Linux, Gentoo, Slackware, Solaris и FreeBSD, а также сборка из исходного кода

Официальный сайт: <https://www.mercurial-scm.org> [2].

Ссылка на скачивание: <https://www.mercurial-scm.org/downloads> [3].

### 1.2.2 Git

Полное название: Git

Разработчик: Тинус Торвальдс и Джунио Хамано

Лицензия: GNU GPL

Первый выпуск: 7 апреля 2005

Последняя версия: 2.10.2 (28 октября 2016)

Платформа для серверной части: GitLab

Платформа для клиентской части: GitKraken, SmartGit, gitk, QGit, Gigggle, gitg, TortoiseGit, SourceTree, Git-cola, GitX, Gitti, Gitbox, Github-клиент, StGit, GitTower

Платформа: Windows, MacOS, Debian, Ubuntu, Fedora, Gentoo, Arch Linux, openSUSE, FreeBSD, Solaris, OpenBSD, Alpine, RedHat Enterprise Linux, Oracle Linux и CentOS, а также сборка из исходного кода

Официальный сайт: <https://git-scm.com/downloads> [4].

Ссылка на скачивание: <https://git-scm.com/downloads> [5].



### 1.2.3 Concurrent Version System

Полное название: Concurrent Versions System

Краткое название: CVS

Разработчик: The CVS Team

Лицензия: GNU GPL

Первый выпуск: 1990

Последняя версия: 1.11.23 (8 мая 2008)

Платформа: Windows, Linux, Mac OS, FreeBSD

Платформы для серверной части: сервер, доступный с официального сайта CVS

Платформы для клиентской части: кроссплатформенные – CVSGUI, LinCVS, TkCVS, TortoiseCVS; Microsoft Windows – TortoiseCVS, WinCVS, DevC++; Linux – Cervisia, pharmacy; Solaris – Sun Studio IDE

Официальный сайт: <http://nongnu.org/cvs> [6].

Ссылка на скачивание: <http://www.nongnu.org/cvs/#downloading> [7].

### 1.2.4 Subversion

Полное название: Subversion

Краткое название: SVN

Разработчик: Apache Software Foundation

Лицензия: Apache License 2.0

Первый выпуск: 20 октября 2000

Последняя версия: 1.9.4 (28 апреля 2016)

Платформа для серверной части: SilkSVN, модуль mod\_dav\_svn к серверу Apache

Платформа для клиентской части: TortoiseSVN, RapidSVN, RabbitSVN, VisualSVN

Платформа: GNU/Linux, Windows, MacOS, AIX, FreeBSD, NetBSD, OpenBSD, HP-UX и Solaris, а также сборка из исходного кода

Официальный сайт: <https://subversion.apache.org> [8].

Ссылка на скачивание: <https://subversion.apache.org/packages.html> [9].

### **1.2.5 Perforce**

Полное название: Perforce

Краткое название: P4

Разработчик: Perforce Software

Лицензия: Проприетарная

Первый выпуск: 1995

Последняя версия: 2016.1/1411799 (14 июня 2016)

Платформа: AIX, FreeBSD, HP-UX, Linux, Mac OS X, NetBSD, Solaris,  
Windows

Платформа для клиентской и серверной частей: Perforce Helix

Официальный сайт: <https://perforce.com/helix/> [10].

Ссылка на скачивание: <https://www.perforce.com/downloads/helix/> [11].

## 2 Mercurial

### 2.1 Краткая характеристика

Распределенная система контроля версий Mercurial разрабатывалась Мэттом Макалом параллельно с системой контроля версий Git, созданной Торвальдсом Линусом. Первоначально, она была создана для эффективного управления большими проектами под Linux'ом, а поэтому была ориентирована на быструю и надежную работу с большими репозиториями. На данный момент Mercurial адаптирован для работы под *Windows*, *Mac OS X* и большинство *Unix* систем [12].

Большая часть системы контроля версий написана на языке Python, и только отдельные участки программы, требующие наибольшего быстродействия, написаны на языке Си. Идентификация ревизий происходит на основе алгоритма хеширования SHA1 (Secure Hash Algorithm 1), однако, также предусмотрена возможность присвоения ревизиям индивидуальных номеров. Так же, как и в git'е, поддерживается возможность создания веток проекта с последующим их слиянием [12].

Для взаимодействия между клиентами используются протоколы *HTTP*, *HTTPS* или *SSH* [12]. Набор команд — простой и интуитивно понятный, во многом схожий с командами Subversion. Так же имеется ряд графических оболочек и доступ к репозиторию через веб-интерфейс. Немаловажным является и наличие утилит, позволяющих импортировать репозитории многих других систем контроля версий. Рассмотрим основные достоинства и недостатки Mercurial.

Mercurial имеет механизм хранения Revlog. Каждому файлу, помещенному в хранилище, сопоставляется два других: индекс и файл с данными. Файлы с данными содержат слепки и дельта-слепки, которые создаются только когда количество отдельных изменений файла превышает некоторое пороговое значение. Индекс служит инструментом эффективного

доступа к файлу с данными. Дельты, полученные в результате изменения файлов под контролем версий, добавляются только в файлы с данными. Для того, чтобы правки из разных мест файла объединить в одну ревизию, используется индекс. Ревизии отдельных файлов складываются манифесты, а из манифестов — фиксации. Этот метод зарекомендовал себя весьма эффективным в деле создания, поиска и вычисления различий в файлах. Также к достоинствам метода можно отнести компактность по отношению к месту на диске и достаточно эффективный протокол передачи изменений по сети.

Наряду с традиционными возможностями систем контроля версий, Mercurial поддерживает полностью децентрализованную работу (отсутствует понятие основного хранилища кода), ветвление, слияние репозиториях. Поддерживается обмен данными между репозиториями через HTTP/HTTPS, SSH и вручную при помощи упакованных наборов изменений.

Утилита Hg обладает компактным интерфейсом, и Mercurial считается более простой в освоении системой, чем, например, Git [12].

## 2.2 Установка клиентской части

На данный момент Mercurial адаптирован для работы под *Windows*, *Mac OS X* и большинство *Unix* систем [3].

Установка под систему:

- Linux можно осуществить через командную строку (`$ sudo apt-get install mercurial`) [3];
- Windows можно осуществить через официальный сайт [3];
- Mac OS X можно осуществить через командную строку (`$ brew install mercurial`) [3].

## 2.3 Основные сценарии использования

Для системы контроля версий Mercurial представлен сценарий необходимый для отображения всех основных возможностей. Для каждого сценария представлена конфигурацию в виде листинга или скриншотов.

Использовать систему Mercurial можно как из command-line (командной строки), так и с помощью GUI (Графического Пользовательского Интерфейса).

### 2.3.1 Сценарии для command-line

Список рассматриваемых сценариев command-line:

- создание нового проекта;

Для инициализации нового репозитория нужно перейти в необходимый каталог, в котором будет располагаться проект и использовать команду *hg init*, где одним из параметров может быть имя проекта [13].

- получение копии существующего репозитория в новом месте;

Для получения копии проекта используют команду *hg copy*, система Mercurial делает копию каждого исходного файла в том виде, в каком он находится в рабочей директории. Если сделать некоторые модификации в файле, и затем сделали *hg copy* без предварительной фиксации этих изменений (commit changes), то новая копия будет также содержать сделанные модификации, которые были сделаны до этой точки.

Для команды *hg copy* необходимо предоставить два или более параметра, в которых последний параметр будет обработан как место назначения (destination), и все остальные параметры как источники (sources).

Для получения клона проекта можно с помощью команды *hg clone*, где после ключевого слова необходимо указать место от куда клонируется и последний параметр определяет каталог назначения [13].

- добавление файлов;

Система Mercurial не будет работать с файлами в репозитории, пока не укажут ей поддерживать эти файлы. Команда **hg status** покажет, о каких файлах Mercurial ничего не знает; для отображения таких файлов используют подкоманду “?”. Чтобы указать Mercurial отслеживать файл, используйте команду **hg add**. Как только добавлены файлы, запись в выводе команды **hg status** поменяется с “?” на “A”. Затем **hg commit** запустит файлы в репозиторий [13].

- добавление папок;

Система Mercurial никак не отслеживает информацию о директориях. Вместо этого она отслеживает путь до файла. Перед тем, как создать файл, Mercurial сначала создает недостающие директории, которые составляют путь до файла. После того, как Mercurial удаляет файл, она затем удаляет любые пустые директории, которые находятся в пути файла.

Один из способов добавления папок: создайте папку, и выполните команду **hg add** для скрытого (hidden) файла в этом каталоге. В Unix-системах принято, что любое имя файла, начинающееся с точки (“.”), считается скрытым и для большинства команд и утилит GUI [13].

Другой способ для сохранения пустых директорий - создание директорий с помощью скрипта автоматической сборки перед тем, как директория потребуется (например, с помощью **makefile**).

- создание ветви разработки из главной ветки в отдельную именованную ветку;

Для создания ветви используется команда **hg branch**, где затем указывается имя ветви [13].

- слияние главной ветки в созданную ветку;

Для слияния главной ветки в созданную необходимо сперва проверить все ли необходимые файлы зафиксированы, иначе изменения потеряются. Затем переходим в именуемую ветку с помощью команды **hg update <имя**

**ветви> -С** в которую будем сливать, затем можно произвести слияние **hg merge** [13].

- слияние именованной ветки в измененную главную ветку;

Для слияния именованной ветки в главную необходимо сперва проверить все ли необходимые файлы закомичены, иначе изменения потеряются. Затем переходим в главную ветвь с помощью команды **hg update <имя ветви> -С** таким образом мы перешли в главную ветвь и теперь можем сливать в нее именованную ветвь командой **hg merge** [13].

- закрытие именованной ветки;

Чтобы закрыть ветку необходимо в консоли ввести **hg commit --close-branch** [13].

- отображение изменения файла.

Для отображения ревизий в консоли вводим команду **hg annotate <имя файла>** в результате получишь список строк, которые были изменены в какой-то ревизии [13].

### 2.3.2 Сценарии для GUI

Данные сценарии будут рассмотрены для TortoiseHg.

Список рассматриваемых сценариев для GUI:

- создание нового проекта;

Для создания нового проекта необходимо перейти по следующему пути «Файл – Новое хранилище». В открывшемся окне (рисунок 2.1) указать путь где необходимо расположить проект, так же можно установить дополнительные настройки, затем нажать «Создать», в результате у вас появится новый проект (рисунок 2.2).

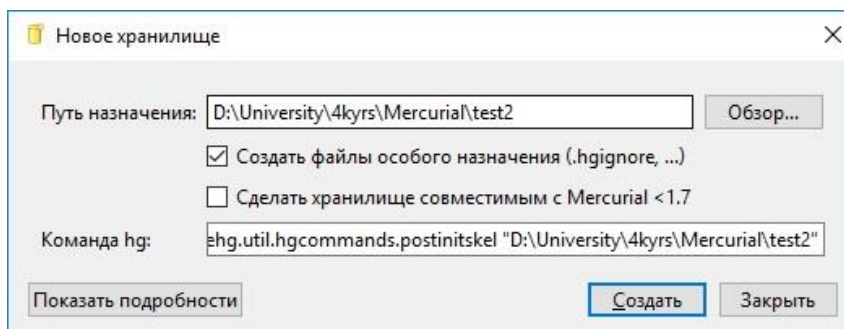


Рисунок 2.1 – Создание нового проекта

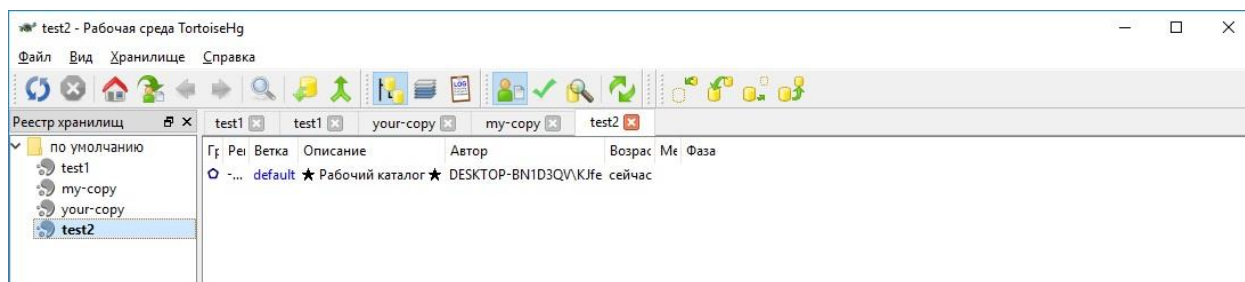


Рисунок 2.2 – Результат созданного репозитория

– получение копии существующего репозитория в новом месте;

Для получения копии существующего репозитория необходимо перейти по следующему пути «Файл – Клонирование». В открывшемся окне (рисунок 2.3) указать путь к источнику, а затем указать путь назначения, так же можно установить дополнительные настройки, затем нажать «Клонировать», в результате у вас появится клонированный проект (рисунок 2.4).

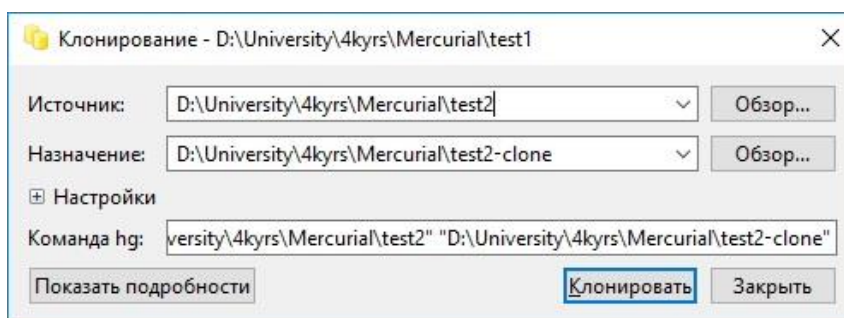


Рисунок 2.3 – Клонирование репозитория

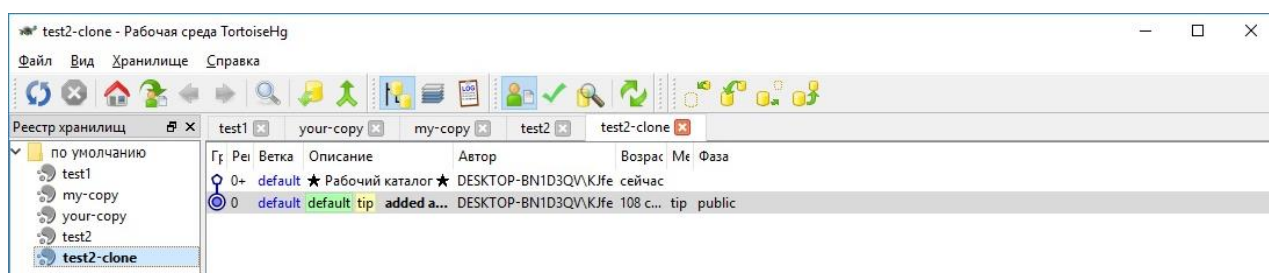




Рисунок 2.4 – Результат клонирования репозитория

– добавление файлов;

Для дополнения файлов в репозиторий необходимо нажать на ярлык  «Фиксация» затем нажать на ярлык  «Обновить изображение списка файлов» система сама определит какое состояние у файла (рисунок 2.5).



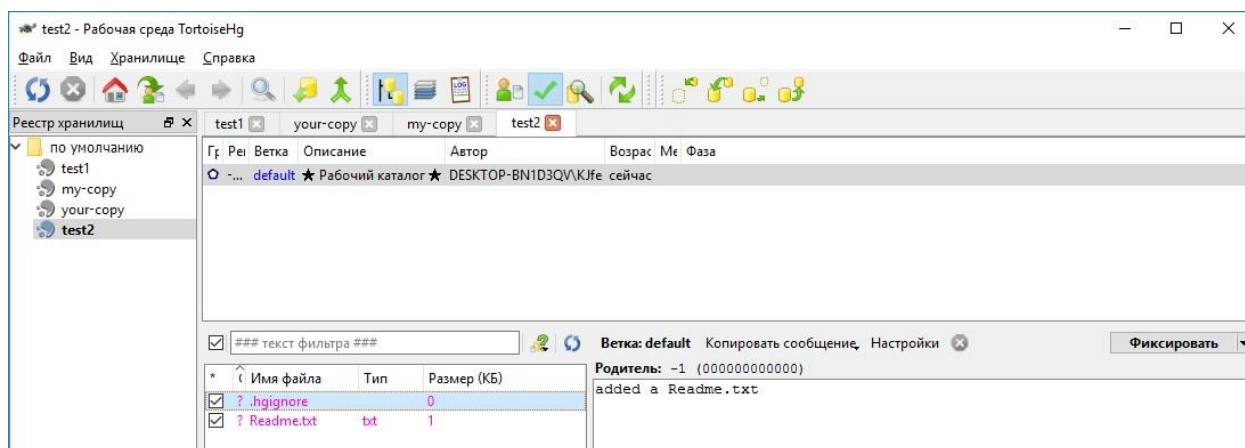


Рисунок 2.5 – Отображение списка измененных файлов

Состояния файлов:

- М – измененный;
- А – добавленный;
- R – удаленные;
- ! – отсутствующие;
- ? – неизвестные;
- I – игнорируемые;
- C – не измененные;
- S – подхранилище.

Для выбранных необходимых файлов, отмеченных галочкой, в окне справа пишем комментарий, который должен отражать суть произошедшего с файлами, это облегчит поиск необходимой фиксации среди их множества и нажать «Фиксировать». В результате получим (рисунок 2.6) зафиксированные файлы в репозитории с их изменениями.

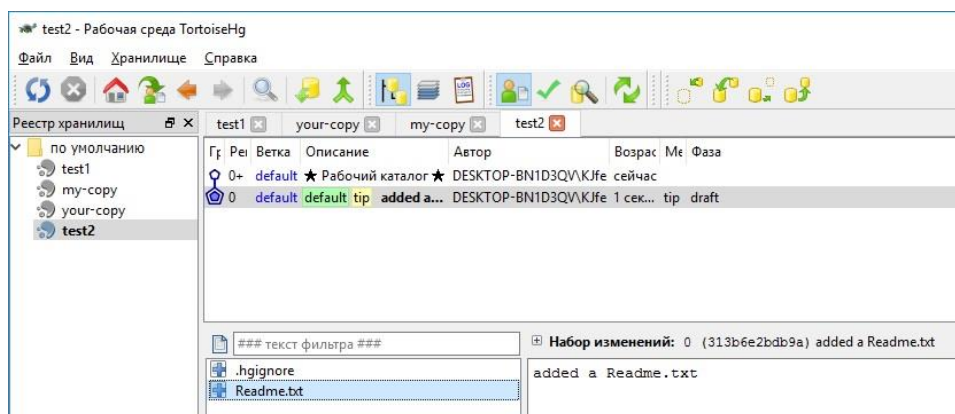



Рисунок 2.6 – Результат добавления файлов

- добавление папок;

Система Mercurial никак не отслеживает информацию о директориях. Вместо этого она отслеживает путь до файла. Перед тем, как создать файл, Mercurial сначала создает недостающие директории, которые составляют путь до файла. После того, как Mercurial удаляет файл, она затем удаляет любые пустые директории, которые находятся в пути файла.

- создание ветви разработки из главной ветки в отдельную именованную ветку;

Для создания из главной ветки именованную ветвь необходимо нажать на ярлык  «Фиксация», затем нажать на кнопку «Ветка: default», где default – начальная ветка. В появившемся окне выбираем «Открыть новую именованную ветку» пишем название этой ветки (рисунок 2.7) и нажимаем «Ок». Затем необходимо зафиксировать создание ветки написав комментарий к фиксации и нажать «Фиксировать», в появившемся окне выбрать необходимое (рисунок 2.8), если в момент создания ветки были выбраны незафиксированные файлы, то можно их зафиксировать в новую ветку или в предыдущую. В результате создана новая именованная ветка из главной (рисунок 2.9).

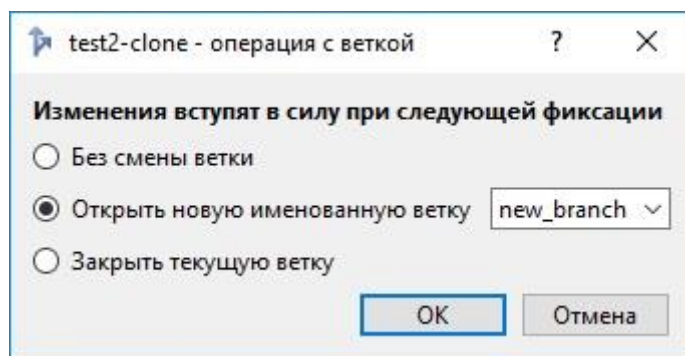


Рисунок 2.7 – Создание новой ветки

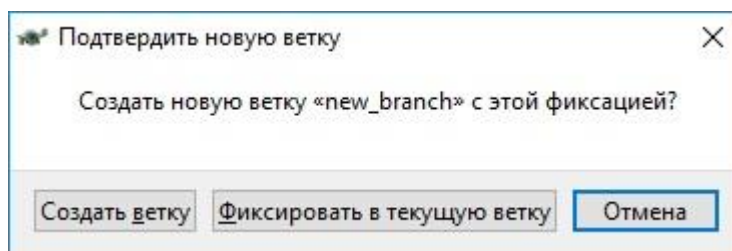


Рисунок 2.8 – Подтверждение создания новой ветки

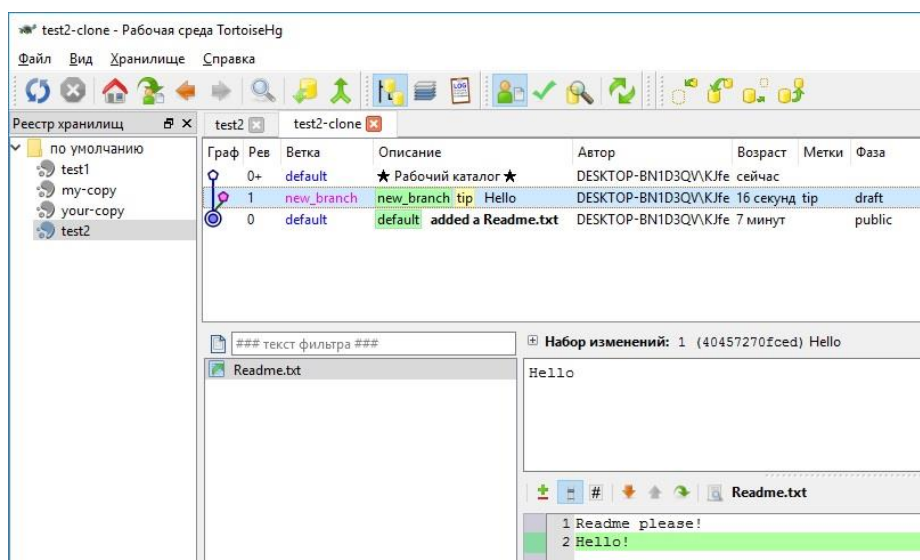


Рисунок 2.9 – Результат создания именованной ветки

- слияние главной ветки в созданную ветку;

Прежде чем сливать ветки необходимо что б все файлы были зафиксированы. Для слияния главной ветки в созданную необходимо сперва перейти в именованную ветку (рисунок 2.10), после этого выполнить «Хранилище – Слить...», в появившемся окне (рисунок 2.11) проверяем направления слияния и параметры нажимаем «Далее», системы пытается произвести слияние автоматически (рисунок 2.12) если нет конфликтов, иначе конфликты нужно разрешить и нажать «Далее». В итоге получаем слитую главную ветку в именованную (рисунок 2.13).

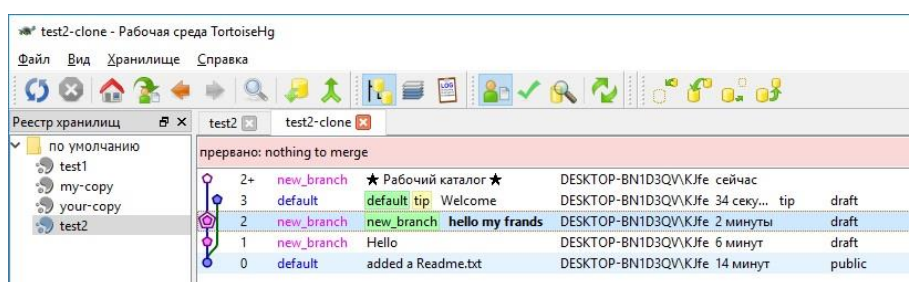


Рисунок 2.10 – Переходим в именованную ветку

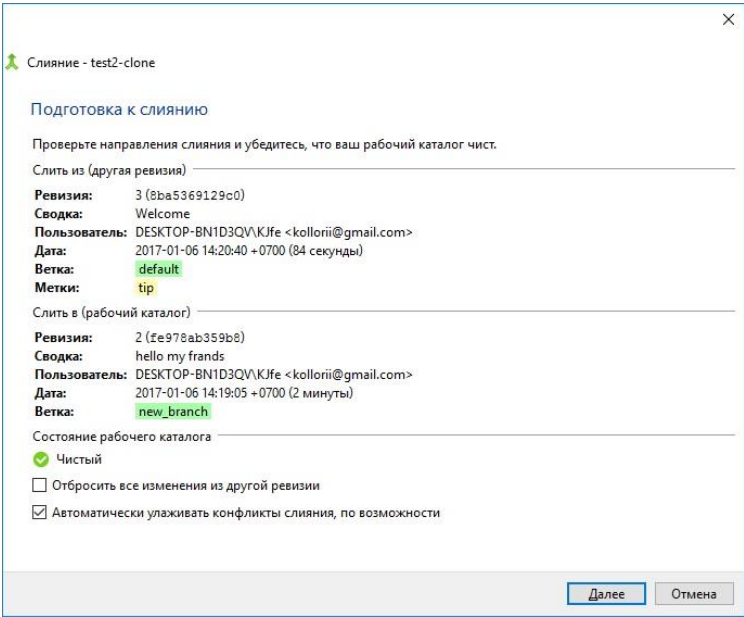


Рисунок 2.11 – Подготовка к сливанию

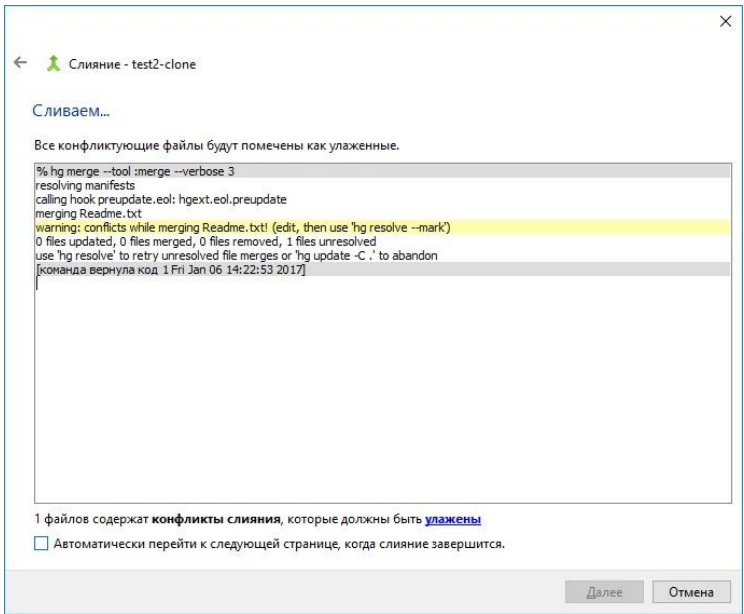


Рисунок 2.12 – Сливание ветвей

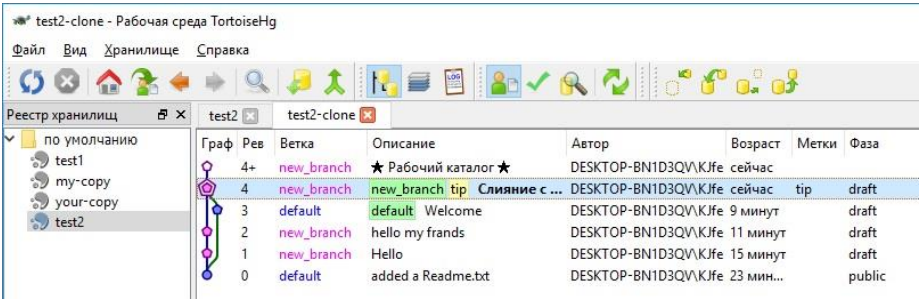


Рисунок 2.13 – результат сливание главной ветки в именованную

– сливание именованной ветки в измененную главную ветку;

Прежде чем сливать ветки необходимо что б все файлы были зафиксированы. Для слияния именованной ветки в главную необходимо сперва перейти в главную ветку (рисунок 2.14), после этого выполнить «Хранилище – Слить...», в появившемся окне (рисунок 2.15) проверяем направления слияния и параметры нажимаем «Далее», системы пытается произвести слияние автоматически (рисунок 2.16) если нет конфликтов, иначе конфликты нужно разрешить.

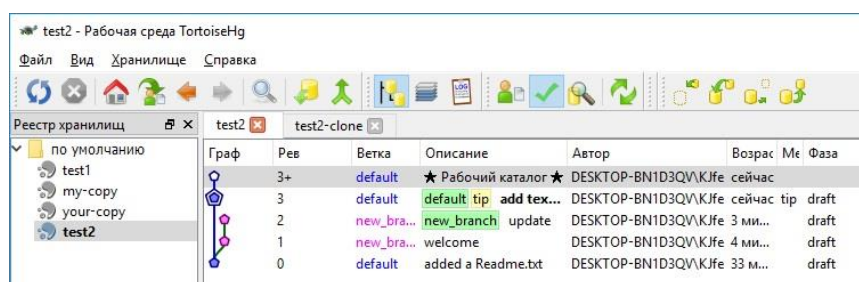


Рисунок 2.14 – Переход в главную ветку

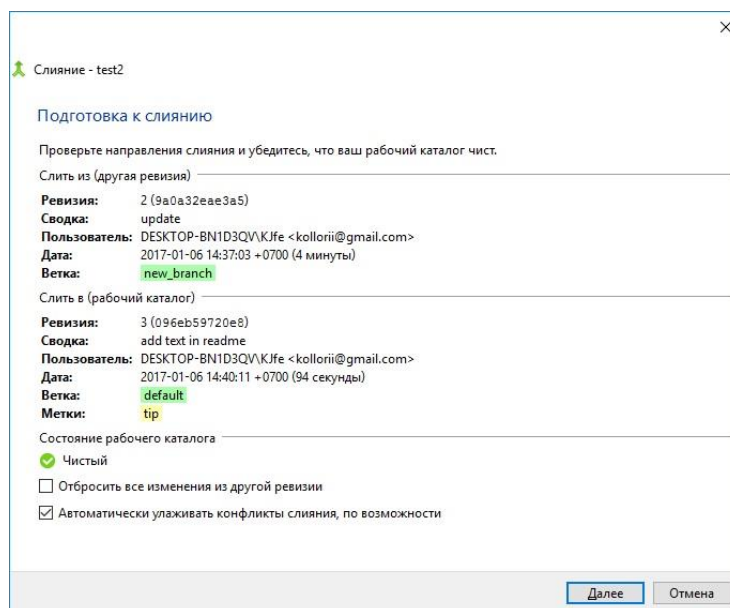


Рисунок 2.15 – Подготовка к сливанию ветвей

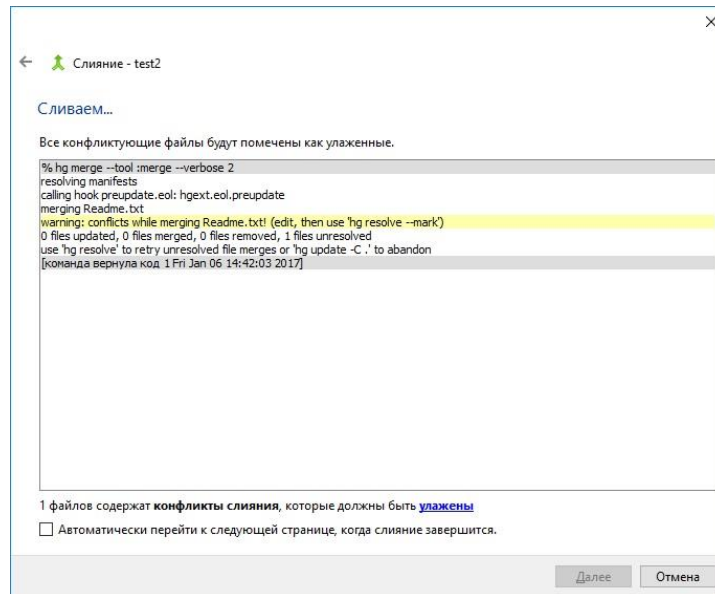


Рисунок 2.16 – Слияние ветвей

Для разрешения конфликтов необходимо нажать на слово улажены, таким образом перейдем в окно где выбираем каким инструментом разрешаем конфликт (рисунок 2.17), выбираем конфликт и справа нажимаем «Уладить средством», открывается окно с четырьмя панелями (рисунок 2.18), где левая верхняя – оригинальный файл, верхняя средняя – версия в именованной ветви, верхняя правая – версия в главной ветви, нижняя – редактор в котором собирается итоговый файл. В результате получаем файл с разрешенным конфликтом.

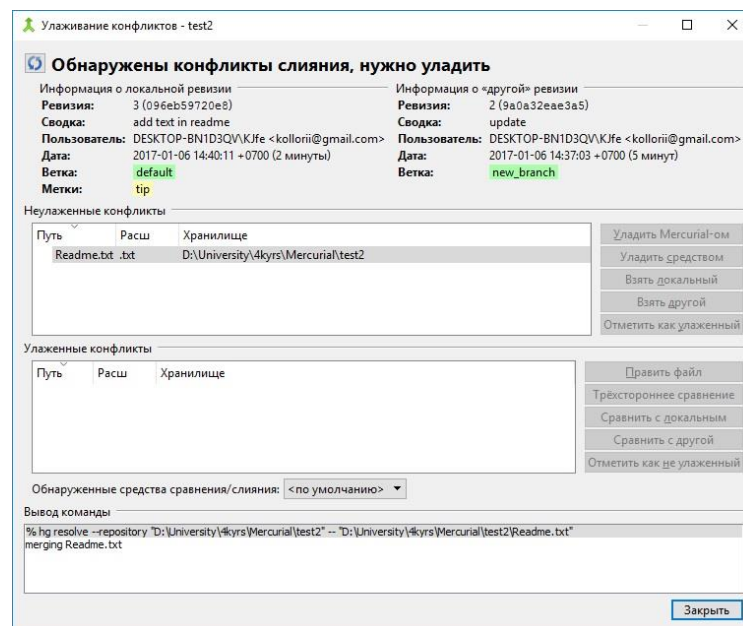


Рисунок 2.17 – Разрешение конфликтов



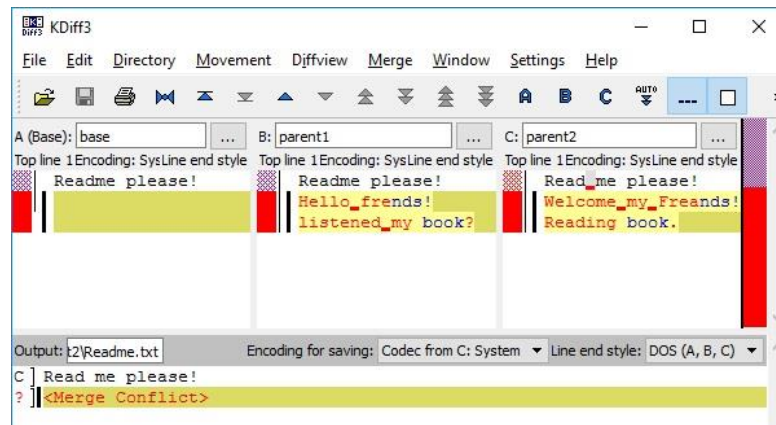


Рисунок 2.18 – Средство разрешения конфликтов KDiff3

- закрытие именованной ветки;

Для закрытия именованной ветки необходимо проделать аналогичные действия что и для создания ветки, только выбрать пункт закрыть ветку (рисунок 2.19).

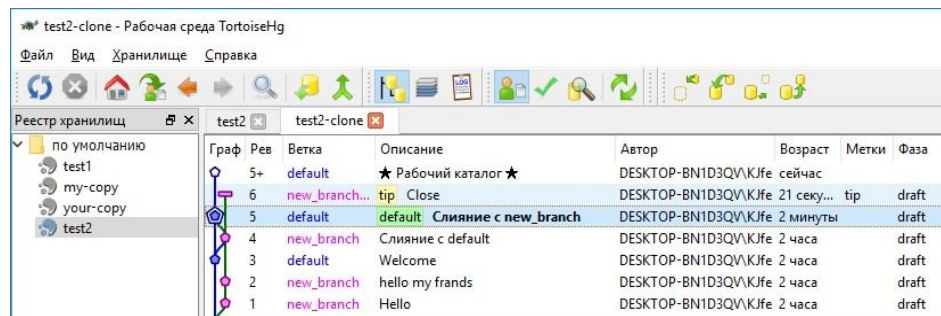



Рисунок 2.19 – Закрытие именованной ветки

- отображение номеров ревизий.

Для отображения изменения файла необходимо выбрать  «Аннотировать номера ревизий» в панели отображения содержимого файлов. В результате появиться столбец с номерами ревизий (рисунок 2.20).

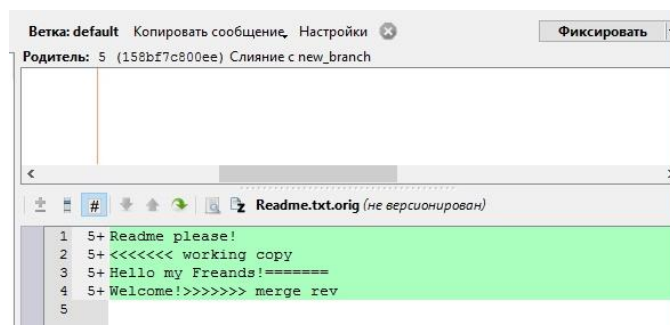


Рисунок 2.20 – Отображения номеров ревизий

## 2.4 Возможности интеграции IDE Visual Studio

Для интеграции IDE Visual Studio в Mercurial существует платформа VisualHg который позволяет работать с системой версионного контроля Mercurial, по функциональности схожа с git, но уступает в практичности.

Название: VisualHg

Текущая версия: 1.1.7 (26 июля 2015г.)

Официальный сайт: <https://visualhg.codeplex.com> [14].

Ссылка на скачивание: <https://visualhg.codeplex.com/releases/view/40544> [15].

Установка осуществляется с помощью стандартного установщика Windows и не представляет ни каких трудностей.

Установка VisualHg в качестве основного плагина версионного контроля представлена на рисунке 2.21.

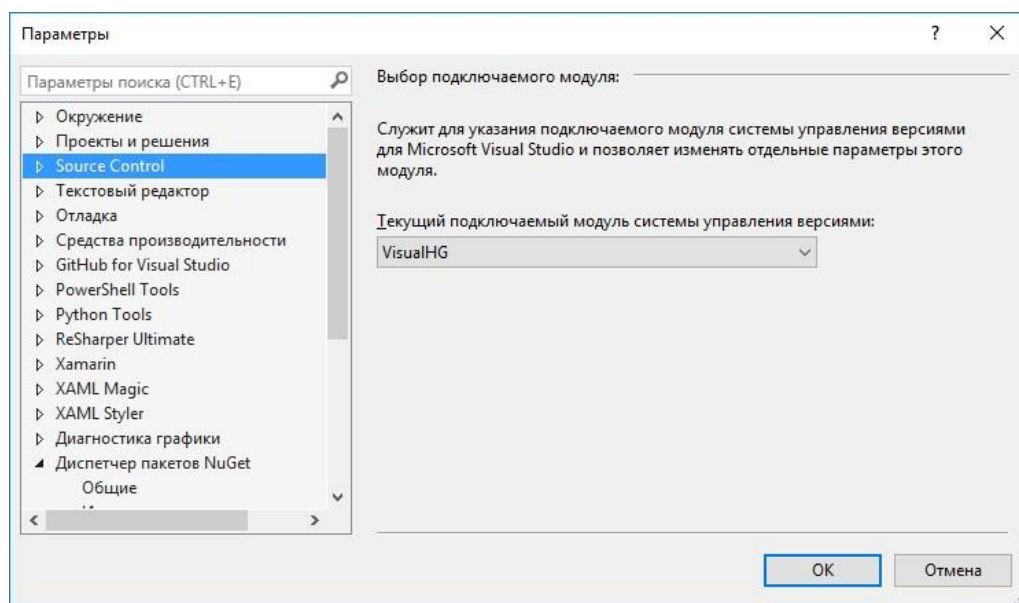


Рисунок 2.21 – Установка плагина VisualHg основным для системы версионного контроля



Создание проекта консольное приложение представлено на рисунке 2.22.

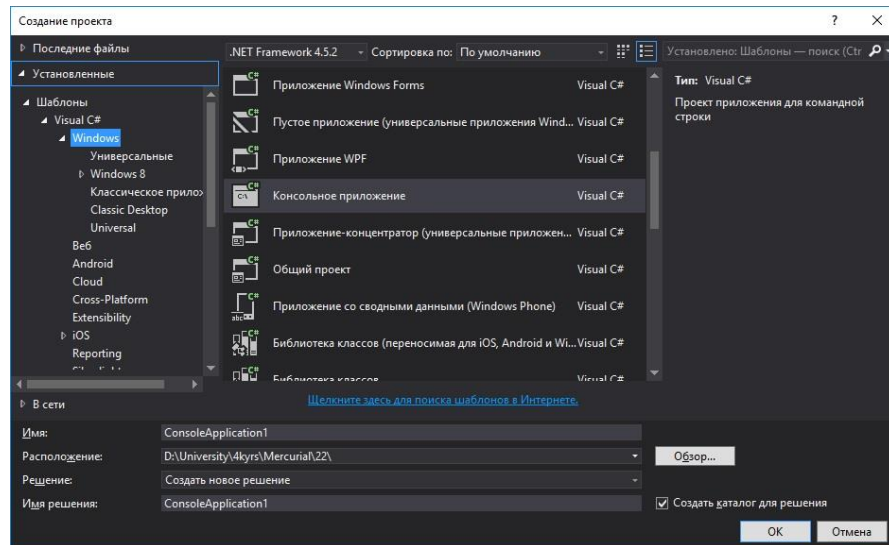


Рисунок 2.22 – Создание проекта

Добавим в проекте вывод «Start» в консоль. Фиксация созданного в проекте вывода строки в консоль представлен на рисунке 2.23. Журнал файла представлена на рисунке 2.24.

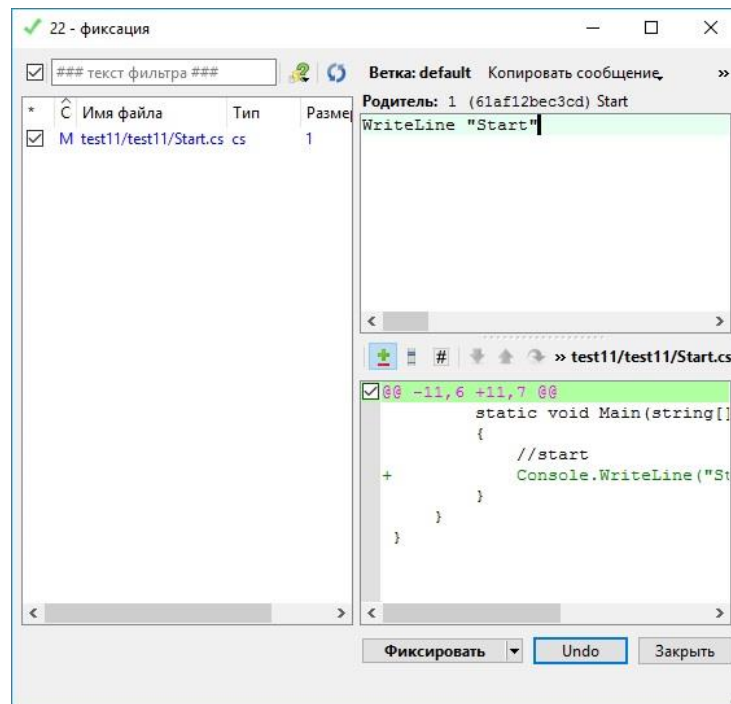


Рисунок 2.23 – Фиксация изменений

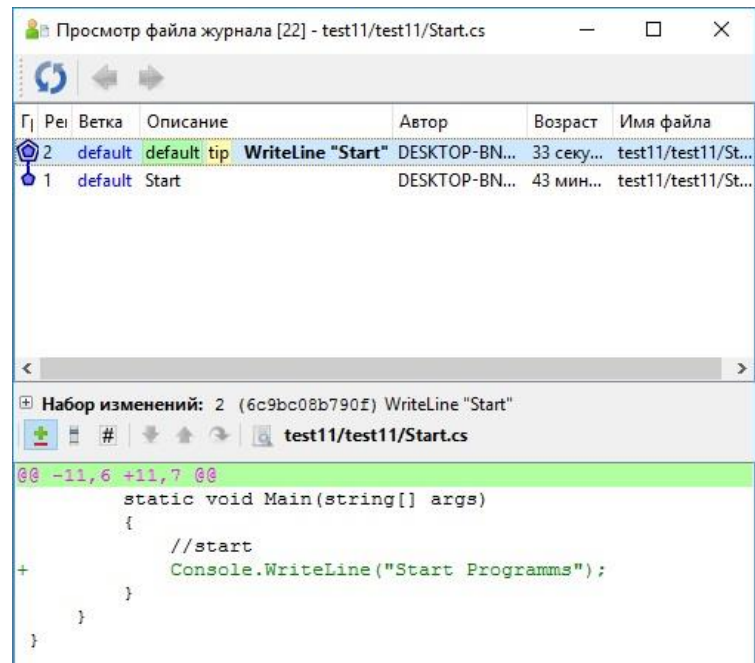


Рисунок 2.24 – Просмотр файла журнала

### 3 Сравнение с другими системами

#### 3.1 Сводная сравнительная таблица

Таблица 3.1 – Сравнение Mercurial с другими системами

Критерий	Subversion	Mercurial	Git	CVS	Perforce
Лицензия	Apache License 2.0	GNU GPL	GNU GPL	GNU GPL	Perforce Perpetual Free License
Установка серверной части	1	3	2	5	4
Установка клиентской части	5	1	3	4	2
Работа с командной строки (CLI)	4	1	3	4	5
Работа с графическим интерфейсом (GUI)	4	3	1	2	5

Таблица 3.2 – Сравнение таблиц со сценариями

Критерий	Subversion	Mercurial	Git	CVS	Perforce
Создание нового проекта	3	5	2	1	4
Получение копии существующего репозитория в новом месте	3	2	1	5	4
Добавление файлов	4	3	2	5	1
Добавление папок	2	3	5	4	1
Создание ветви из главной ветки в отдельную именованную ветку	3	2	1	5	4
Слияние главной ветки в созданную ветку	3	1	2	5	4
Слияние именованной ветки в измененную главную ветку	3	1	2	5	4
Закрытие именованной ветки	5	2	1	4	3
отображение номеров ревизий	5	4	2	3	1

### 3.2 Выводы из сравнения

На основании сравнения систем версионного контроля сделаны следующие выводы.

Blame в CVS интуитивно ясен, однако базовые операции, такие как работа с файлами, папками, ветвление, а также фиксации, сопряжены со сложностями. Работа с графическим интерфейсом в CVS не вызывает трудностей.

Работа с файлами и папками в Subversion сопряжены с некоторыми сложностями, однако такие операции, как ветвление и фиксации, легко осуществимы. Установка серверной части в Subversion сопряжена со сложностями, работа через интеграцию не вызывает трудностей.

Добавление папок в Git и Mercurial может вызывать трудности, другие базовые операции легко осуществимы. Установка клиентской части, а также работа с графическим интерфейсом и интерфейсом командной строки в Git не вызывает трудностей.

Установка серверной части сопряжена с трудностями. Создание нового проекта в Mercurial вызывает трудности, Blame не совсем интуитивно ясен; остальные операции легко осуществимы. Работа с графическим интерфейсом и интерфейсом командной строки в Mercurial не вызывает трудностей.

Работа с файлами и папками в Perforce интуитивно ясна, остальные операции сопряжены со сложностями. Работа с графическим интерфейсом и интерфейсом командной строки в Perforce сопряжена со сложностями.

## 4 Заключение

Были рассмотрены такие системы контроля версий, как Subversion, Mercurial, Concurrent Version System, Git, Perforce. Были рассмотрены основные сценарии использования системы контроля версий Subversion, такие как создание нового проекта, добавление файлов и папок, операции ветвления, просмотр истории версий файлов.

На основании сравнения вышеописанных систем остальными участниками проектной группы была создана сводная сравнительная таблица основных сценариев использования, а также сводная сравнительная таблица характеристик систем. На основании последних был сделан вывод о том, что наиболее лучшей системой контроля версий по всем рассматриваемым сценариям использования является система контроля версий *Git*.

## Список использованных источников

- 1 Система управления версиями [Электронный ресурс]: Wikipedia – Режим доступа: [https://ru.wikipedia.org/wiki/Система\\_управления\\_версиями](https://ru.wikipedia.org/wiki/Система_управления_версиями) (дата обращения 28.12.16)
- 2 Официальный сайт Mercurial [Электронный ресурс]: Mercurial. – Режим доступа: <https://www.mercurial-scm.org> (дата обращения 29.12.16)
- 3 Get started [Электронный ресурс]: Mercurial. – Режим доступа: <https://www.mercurial-scm.org/downloads> (дата обращения 29.12.16)
- 4 Официальный сайт Git [Электронный ресурс]: Git. – Режим доступа: <https://git-scm.com/> (дата обращения 29.12.16)
- 5 Downloads [Электронный ресурс]: Git. – Режим доступа: <https://git-scm.com/downloads> (дата обращения 29.12.16)
- 6 Официальный сайт CVS [Электронный ресурс]: CVS. – Режим доступа: <http://nongnu.org/cvs> (дата обращения 29.12.16)
- 7 Downloads [Электронный ресурс]: CVS. – Режим доступа: <http://www.nongnu.org/cvs/#downloading> (дата обращения 29.12.16)
- 8 Официальный сайт Subversion [Электронный ресурс]: Subversion. – Режим доступа: <https://subversion.apache.org> (дата обращения 29.12.16)
- 9 Downloads [Электронный ресурс]: Subversion. – Режим доступа: <https://subversion.apache.org/packages.html> (дата обращения 29.12.16)
- 10 Официальный сайт Perforce [Электронный ресурс]: Perforce. – Режим доступа: <https://perforce.com/helix/> (дата обращения 29.12.16)
- 11 Downloads [Электронный ресурс]: Perforce. – Режим доступа: <https://www.perforce.com/downloads/helix/> (дата обращения 29.12.16)
- 12 Описание [Электронный ресурс]: Wikipedia. – Режим доступа: <https://ru.wikipedia.org/wiki/Mercurial> (дата обращения 29.12.16)
- 13 Learning Mercurial in Workflows [Электронный ресурс]: Mercurial. – Режим доступа: <https://www.mercurial-scm.org/guide> (дата обращения 30.12.16)

14 Официальный сайт VisualHg [Электронный ресурс]: CodePlex. – Режим доступа: <https://visualhg.codeplex.com> (дата обращения 30.12.16)

15 Downloads [Электронный ресурс]: CodePlex. – Режим доступа: <https://visualhg.codeplex.com/releases/view/40544> (дата обращения 30.12.16)