**«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»**

**(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Факультет |  | И |  | Информационных и управляющих систем |
|  |  | шифр |  | наименование |
| Кафедра |  | И5 |  | Систем управления и компьютерных технологий |
|  |  | шифр |  | наименование |
| Дисциплина |  | Проблемы человеко-машинного взаимодействия | | |

КУРСОВАЯ РАБОТА

на тему

|  |
| --- |
|  |
| Построение способов взаимодействия пользователя с |
| разрабатываем программно-аппаратным комплексом |

Вариант 21

Система контроля версий документов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | | | |  | | И9М31 |
| Разомазов А.В. | | | | | | |
| Фамилия И.О. | | | | | | |
| **РУКОВОДИТЕЛЬ** | | | | | | |
| Гущин А.Н. | |  |  | | | |
| Фамилия И.О. Подпись | | | | | | |
| Оценка |  | | | |  | |
| «\_\_\_\_\_» |  | | | | 20\_\_\_\_ г. | |

САНКТ-ПЕТЕРБУРГ

2018 г.

СОДЕРЖАНИЕ

[ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 3](#_Toc534840353)

[ВВЕДЕНИЕ 4](#_Toc534840354)

[1 АНАЛИЗ ЦЕЛЕВОЙ АУДИТОРИИ 5](#_Toc534840355)

[1.1 ОПРЕДЕЛЕНИЕ ЦЕЛЕВОЙ АУДИТОРИИ 5](#_Toc534840356)

[1.2 ТРЕБОВАНИЯ К ПРОЕКТИРОЕМОМУ ПРИЛОЖЕНИЮ 5](#_Toc534840357)

[2 ВЫБОР АППАРАТНО-ПРОГРАММНОЙ ПЛАТФОРМЫ 7](#_Toc534840358)

[3 ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 8](#_Toc534840359)

[3.1 ОБЩИЕ ТРЕБОВАНИЯ К ПОЛЬЗОВАТЕЛЬСКОМУ ИНТЕРФЕЙСУ 8](#_Toc534840360)

[3.2 ПРОТОТИП ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 10](#_Toc534840361)

[3.3 СРАВНЕНИЕ С БЛИЗКИМИ ПО ФУНКЦИОНАЛЬНОСТИ ПРОГРАММНЫМИ ПРОДУКТАМИ 13](#_Toc534840362)

[3.4 ОЦЕНКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 15](#_Toc534840363)

[ЗАКЛЮЧЕНИЕ 17](#_Toc534840364)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc534840365)

# ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ПО – программное обеспечение

ВУЗ – высшее учебное заведение

ПК – персональный компьютер

СКВ – система контроля версий

# ВВЕДЕНИЕ

Студенты технических учебных заведений практически постоянно работают с документами, такими как тексты, рисунки, чертежи, изменяя их по несколько раз в короткий промежуток времени. Порой получается так, что правки, сделанные в течение последней недели, необходимо отменить и вернуться к документам в состоянии недельной давности. Хорошо, если правок было сделано немного, в этом случае может помочь правка документа средствами редактора. Однако если в течение этой недели шла более-менее активная работа с документом, просто так восстановить статус «до важной правки, сделанной неделю назад» не получится. Для этого необходима копия документа на момент «до важной правки», а также еще десяток копий «до другой важной правки», «до сомнительной правки» и «до правки, которую, скорее всего, придется отменить». В принципе, такой подход возможен и практикуется многими. Чаще всего пользователи держат важные версии файлов, сохраняя их с префиксами «дата\_время». Преимуществом этого метода является простота, недостатком – неудобство использования и огромное количество одних и тех же файлов. И, если со второй проблемой можно как-то бороться, то с неудобством необходимо что-то делать.

Для решения подобных проблем были созданы системы управления версиями (или системы контроля версий) – 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.

Главной целью выполнения данной курсовой работы является определение ЦА в зависимости от сферы применения и особенностей пользователей, анализ функциональности СУВ исходя из потребностей, требований и особенностей пользователей, а также макетирование пользовательского интерфейса для приложения.

# 1 АНАЛИЗ ЦЕЛЕВОЙ АУДИТОРИИ

## ОПРЕДЕЛЕНИЕ ЦЕЛЕВОЙ АУДИТОРИИ

Система контроля версий - система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов [1]. СКВ можно применять практически в любой сфере деятельности, связанной с вычислительной техникой. Это решение больше всего подходит тем пользователям, кто работает с текстовым документами, программирует или занимается редактированием и обработкой изображений. Большинство редактируемых на ПК документов можно добавить в версионный контроль. Так как в современном мире вся документация готовится в электронном виде, эти системы очень актуальны и распространены [2]. Сразу определим, что подобные системы можно применять и при индивидуальной работе на одном ПК, и при работе в группах с использованием сервера, что является главным фактором в определении целевой аудитории ПО.

Исходя из вышесказанного, определим группы по типу применения программы:

* индивидуальное применение (писатели, проектировщики, программисты, преподаватели, студенты);
* групповое применение. (Научные центры, конструкторские бюро, отдел закупок).

## 1.2 ТРЕБОВАНИЯ К ПРОЕКТИРОЕМОМУ ПРИЛОЖЕНИЮ

Примем, что программа обладаем стандартным набором функций, которые пригодятся как при индивидуальной работе, так и работе в группе, а, следовательно, разрабатываемое приложение подойдет двум указанным группам и различаться будет лишь организацией работы. Для формирования требований к пользовательскому интерфейсу, прежде всего, необходимо обозначить функциональные требования, которые выдвигаются к СВУ. Стоит сразу отметить, что требования к ПО для работы на предприятиях будут более серьезными, но в целом различия будут только в организации сервера и некоторых других параметров приложения [3].

Проектируемое приложение должно обладать следующими функциями:

* создание рабочих копий документов;
* создание разных вариантов одного документа, то есть ветки с общей историей изменений до точки ветвления и с разными — после неё;
* возможность узнать, кто и когда добавил или изменил конкретный набор строк в файле;
* вести журнал изменений, в который пользователи могут записывать пояснения о том, что и почему они изменили в данной версии и оставлять свои комментарии;
* контролировать права доступа пользователей, разрешая или запрещая чтение или изменение данных, в зависимости от того, кто запрашивает это действие;
* возможность заблокировать, скрыть, переместить, клонировать файл в хранилище;
* возможность объединить изменения, сделанные разными разработчиками (для текстовых файлов);
* возможность работы с текстовыми документами, изображениями, медиа файлами, а так же возможность работы с файлами «уникального» формата (чертежи, 3D модели, расчётные проекты);
* наличие автосохранения и возможность его настройки.

СУВ желательно использовать централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями.

Основным нефункциональным требованием можно считать автоматическую синхронизацию с сервером, а также ее настройка [4].

# 2 ВЫБОР АППАРАТНО-ПРОГРАММНОЙ ПЛАТФОРМЫ

Под программно-аппаратным комплексом подразумевается набор технических и программных средств, работающих совместно для выполнения одной или нескольких сходных задач.

Требования к аппаратно-программной платформе выдвигаются в соответствии поставленным функциональным и нефункциональным требованиям.

Системы должны будут обеспечить следующие функции:

* Сервер:
  + Хранение всех версий файлов на сервере (в т.н. репозитарии), с историей их правок
  + Управление правами доступа, проверка авторизации
* Клиентское приложение:
  + Подключение к серверу, синхронизация файлов с рабочей копией (в обе стороны), отображение истории правок, откат к предыдущей/сохраненной версии, визуальное сравнение версий файлов.

В качестве аппаратной платформы будут рассматриваться ПК, под управлением ОС Windows не ниже 7 версии, а также под управлением Linux.

В качестве сервера можно использовать:

* централизованную систему, для выполнения большинства операций (например, просмотр истории) требуется подключение. Можно назначать права доступа на отдельные подпапки репозитория, брать в работу часть репозитория.
* децентрализованная система, часть операций (например, просмотр истории) выполняется локально. Права доступа можно выдавать выполнять только целиком на весь проект, взятие в работу также выгружает только весь проект. Также децентрализованность означает, что рабочая копия содержит также и всю историю правок (что увеличивает ее размер на локальном диске), и операции синхронизации выполняются не в 1 операцию (*commit*), а в 2 (*commit*-*push*) [5].

Выбор типа сервера зависит от поставленных задач и организации работы.

Для минимальной настройки СКВ на компьютере необходимо задать глобальные параметры, которые будут применяться к вносимым изменениям и подписывать их. Они будут указывать на Вас в истории изменения файла в удаленных репозиториях. Такими глобальными настройками являются имя пользователя и его email. Их можно установить командами в консоли. Все параметры будут помещены в файл с настройками, расположенным в домашнем каталоге пользователя.

# 3 ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

## 3.1 ОБЩИЕ ТРЕБОВАНИЯ К ПОЛЬЗОВАТЕЛЬСКОМУ ИНТЕРФЕЙСУ

Сформулируем некоторые основные требования к пользовательскому

интерфейсу приложения.

1. Соответствие функциональным требованиям. Пользовательский интерфейс в первую очередь должен обеспечивать описанные функциональные требования (раздел 1.2), то есть соответствовать задачам и целям пользователя.
2. Обеспечение высокой скорости работы пользователя. Для реализации этого требования необходимо делать элементы управления заметными и понятными. Названия (в качестве названий элементов можно использовать изображения) элементов должны соответствовать выполняемым функциям.
3. Понятность и логичность. При проектировании пользовательского интерфейса, необходимо помнить о таких вещах, как соответствие названий пунктов меню, кнопок и других элементов их реальным функциям. Для удобства работы пользователя, при проектировании пользовательского интерфейса, необходимо учитывать визуальное разграничение информационных разделов приложения
4. Обеспечение защиты от человеческих ошибок. Для уменьшения количества ошибок пользователя, необходимо помнить о размерах (например, не делать кнопки слишком маленькими), делать четкими границы между визуальными объектами, избегать создания элементов управления, функции которых могут меняться в зависимости от контекста. Так же необходимо заполнять паузы между событиями.
5. Помимо прочего, интерфейс должен быть визуально привлекательным. Привлекательный пользователю интерфейс делает работу с ним более приятной. Но не стоит забывать, что эстетическое восприятие — вещь субъективная. Должен соблюдаться общий стиль оформления элементов интерфейса, единообразие и согласованность формы, размера и других параметров.

Тяжелее добиться, чтобы интерфейс хорошо воспринимался пользователями в среднем – простое внесение исправлений, подсказанных одними пользователями, может отрицательно сказаться на оценках других. Для субъективного удовлетворения пользователей, при разработке интерфейса, необходимо придерживаться следующих общепринятых принципов [6]:

* Принцип повторного использования. Следует стараться использовать многократно внутренние и внешние компоненты, обеспечивая тем самым унифицированность интерфейса и сходство между его похожими элементами.
* Принцип толерантности. Интерфейс должен быть терпимым к ошибкам пользователя. Ущерб от ошибок должен снижаться за счет возможности отмены и повтора действий, и за счёт разумной интерпретации действий пользователя и введённых им данных.
* Принцип обратной связи. Пользователь должен получать сообщения о важных событиях в системе. Сообщения должны быть информативными, краткими, однозначными и написанными на языке, понятном пользователю.
* Принцип структуризации. Пользовательский интерфейс должен быть целесообразно структурирован. Близкие по смыслу, родственные его части должны быть связаны видимым образом, а независимые – разделены.

## 3.2 ПРОТОТИП ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

На рисунке 1 и рисунке 2 представлены прототипы экрана пользовательского интерфейса приложения СВУ. Интерфейс выполнен на основе выдвинутых общих требований к интерфейсу приложения, с некоторыми исключениями (раздел 3.1) и позволяет выполнять действия, соответствующие обозначенным в разделе 1.2 функциональным требованиям к приложению.

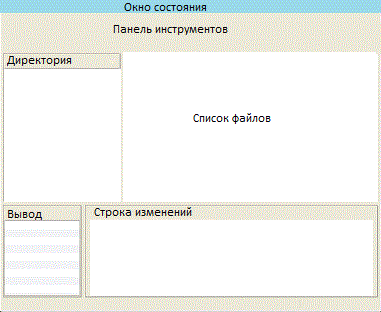


Рисунок 1 – макет пользовательского интерфейса

Интерфейс должен быть простым и интуитивно понятным. Индивидуальная настройка в данном приложении не обязательна ввиду исполнения вспомогательных функций при работе.

Основную область занимает окно со списком файлов в конкретной директории.

Панель инструментов содержит стандартный набор основных необходимых функций, таких как: обновление, удаление, свойства, копирование/вставка файла, ссылка на информацию о документе и так далее с возможностью добавления/удаления дополнительной категории инструментов (рисунок 2).

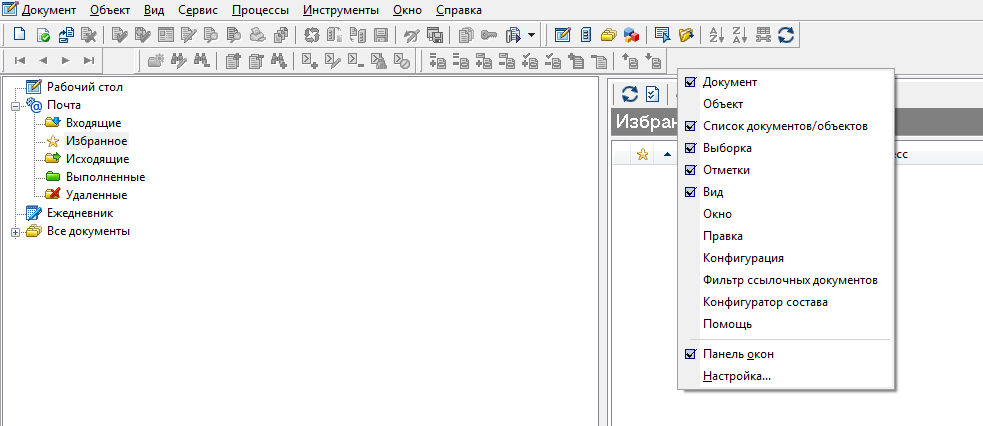


Рисунок 2 – пример вида панели инструментов.

В окне «Директория» выбираются архивы. Есть возможность вывести на первые места «избранные» архивы и скрыть ненужные, а также добавить «выборки» по числу, создателю/владельцу файла, типу, имени и так далее. Желательно наличие функции «Рабочий стол» - автоматически создаваемая выборка документов, с которыми работает/работал пользователь.

Список файлов должен обладать функциями масштабирования, поиска, сортировки по индексу, дате, имени, размеру, а также функции создания и обновления файла. «Вывод» напрямую привязан к списку файлов, выводя все имеющиеся версии, сортированные по времени последнего изменения. В таблице «Строка изменений» выводится информация с комментарием (при его наличии) последнего пользователя, редактировавшего документ.

Основная функциональность построена по типу электронного архива, отсутствие дополнительных функций по типу вкладок, расширенных инструментов или настроек значительно упрощает работу с СУВ, уменьшает нагрузки на систему, выполняя при этом все поставленные задачи.

Данные решения приняты исходя из анализа целевой аудитории. Часть пользователей системы контроля версий документов не будут обладать профессиональными навыками работы с вычислительной техникой (в данном случае с ПК). Эта проблема решается упрощением и возможностью индивидуальной настройки пользовательского интерфейса без потери функциональности программного обеспечения.

## 3.3 СРАВНЕНИЕ С БЛИЗКИМИ ПО ФУНКЦИОНАЛЬНОСТИ ПРОГРАММНЫМИ ПРОДУКТАМИ

Проектируемая в данной работе система СКВ объединяет функции и интерфейсные решения таких программных продуктов (облачных сервисов), как CVS, Git и Mercurial.

Система управления параллельными версиями (Concurrent Versions System) – логическое развитие системы управления пересмотрами версий (RCS), использующая ее стандарты и алгоритмы по управлению версиями, но значительно более функциональная, и позволяющая работать не только с отдельными файлами, но и с целыми проектами. CVS основана на технологии клиент-сервер, взаимодействующих по сети. Клиент и сервер также могут располагаться на одной машине, если над проектом работает только один человек, или требуется вести локальный контроль версий [8].

Главными недостатками являются:

* при перемещении или переименовании файла или директории теряются все, привязанные к этому файлу или директории, изменения.
* сложности при ведении нескольких параллельных веток одного и того же проекта.
* ресурсоемкие операции, так как требуют частого обращения к репозиторию, и сохраняемые копии имеют некоторую избыточность.

Git – это гибкая, распределенная (без единого сервера) система контроля версий, дающая массу возможностей не только разработчикам программных продуктов, но и писателям для изменения, дополнения и отслеживания изменения «рукописей» и сюжетных линий, и учителям для корректировки и развития курса лекций, и администраторам для ведения документации, и для многих других направлений, требующих управления историей изменений. Надежная система сравнения ревизий и проверки корректности данных, основанные на алгоритме хеширования SHA1 (Secure Hash Algorithm 1), гибкая система ветвления проектов и слияния веток между собой и наличие локального репозитория, содержащего полную информацию обо всех изменениях, позволяет вести полноценный локальный контроль версий делает эту систему одной из лучших при работе с ПО [8].

Основные минусы системы:

* Unix – ориентированность. На данный момент отсутствует зрелая реализация Git, совместимая с другими операционными системами.
* возможные (но чрезвычайно низкие) совпадения хеш - кода отличных по содержанию ревизий.
* не отслеживается изменение отдельных файлов, а только всего проекта целиком, что может быть неудобно при работе с большими проектами, содержащими множество несвязных файлов.

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, 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. Основные преимущества:

* быстрая обработка данных;
* кроссплатформенная поддержка;
* возможность работы с несколькими ветками проекта.

Недостатками являются:

* возможные (но чрезвычайно низкие) совпадения хеш - кода отличных по содержанию ревизий;
* ориентирован на работу в консоли.

## 3.4 ОЦЕНКА ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Очевидно, что пользовательский интерфейс напрямую зависит от решаемых задач и, хотя оценка качества пользовательского интерфейса процесс достаточно субъективный и трудно формализуемый, можно с уверенностью утверждать, что хороший интерфейс должен обеспечивать эффективную, быструю и производительную работу пользователя.

На процесс проектирования пользовательского интерфейса набольшее влияние оказывают субъективные представления проектировщика о логичности, удобстве и красоте. Поэтому, большое значение имеет проблема оценки качества пользовательского интерфейса[9].

При тестировании могут учитываться следующие критерии:

* минимальное время на выполнение задач пользователем;
* минимальное количество ошибок, которые допускает пользователь при работе с приложением;
* полное понимание интерфейса пользователями и отсутствие неоднозначностей при работе с ним;
* минимальный объем вводимой пользователями информации;
* простота и визуальная привлекательность интерфейса.

На этапе тестирования прототипа интерфейса может применяться метод фокус-группы для сбора и анализа первоначальных мнений об интерфейсе и доработке элементов, которые вызывают вопросы, а также на этапе тестирования интерфейса рабочего приложения, по результатам которого предлагается список рекомендаций по его улучшению.

Проведение фокус-групп позволяет лучше понять особенности поведения конечных пользователей, что невозможно достичь другими методами, так как пользователи входящие в состав фокус-группы, как правило, являются потенциальными клиентами[10]. В ходе этого процесса измеряется затраченное пользователем время, количество обращений за помощью, ошибки пользователя, его вопросы и комментарии[11].

Для выявления недостатков пользовательского интерфейса проектируемой системы перед ближайшими аналогами, предлагается производить тестирование в сравнении с другими системами, например, Git, SVN, Mercurial.

На основе данных, полученных в результате тестирования пользовательского интерфейса на разных этапах разработки ПО, производится дальнейшая работа по улучшению пользовательского интерфейса.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была обозначена целевая аудитория системы контроля версий, с учетом которой рассмотрены особенности аппаратно-программной платформы, на которой будет базирована система, сформулированы функциональные требования и требования к пользовательскому интерфейсу приложения. Кроме того, был предложен прототип пользовательского интерфейса и определен подход к оценке пользовательского интерфейса проектируемого программного продукта.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Системы управления версиями. [Электронный ресурс]. URL: https://habr.com/post/122060/ (дата обращения: 13.11.2018)
2. Роль системы электронного документооборота для современного предприятия [Электронный ресурс]. URL: http://edrj.ru/article/38-05-17 (дата обращения: 15.10.2018)
3. Все о системах контроля версий. [Электронный ресурс]. URL: http://all-ht.ru/inf/prog/p\_0\_0.html/ (дата обращения: 15.10.2018)
4. Установка и настройка SVN. [Электронный ресурс]. URL: https://habr.com/post/29440/ (дата обращения: 13.11.2018)
5. Орлов С.А. Программная инженерия. Технологии разработки программного обеспечения. 5-е издание СПБ.: Питер, 2016. – 640с.
6. Кенин А. Самоучитель системного администратора. 2-е издание СПБ.:, 2008. – 560с.
7. Основы VCS. [Электронный ресурс]. URL: https://ru.hexlet.io/courses/git\_base/lessons/vcs\_intro/theory\_unit (дата обращения: 15.10.2018)
8. Система контроля версий (CYS) — Сравниваем: Git, SVN, Mercurial URL:https://biz30.timedoctor.com/ru/cystopic10232/ (дата обращения: 20.11.2018)
9. Тестирование пользовательского интерфейса. [Электронный ресурс]. URL: http://www.intuit.ru/studies/courses/1040/209/lecture/5418 (дата обращения: 02.12.2018)
10. Проведение фокус-групп в рамках тестирования UI. [Электронный ресурс]. URL: http://xbsoftware.ru/testirovanie-po/polnij-tsykl/testirovanie-ui/fokus-grupy/ (дата обращения: 02.12.2018)
11. Джеф Раскин, Интерфейс: новые направления в проектировании компьютерных систем. – СПб: «Символ-плюс», 2003. – 272с.