**Глава I**

**Предпроектное исследование сред разработки программного обеспечения**

**Предпроектное исследование сред разработки программного обеспечения**

# 1. Описание предметной области

Кодирование - неотъемлемая часть процесса разработки программного

обеспечения, наряду с такими этапами как проектирование, тестирование, сопровождение.

Продуктивность коллектива разработчиков напрямую зависит от среды

разработки(IDE). Современные среды разработки обладают большим функционалом и количеством различных настроек, благодаря которым пользователи могут настроить свою среду оптимальным для себя образом.

Часть IDE ориентирована на использование в конкретной операционной системе, например, такие среды, как MS Visual Studio (Windows), Xcode (MacOS), а часть является кроссплатформенными - PhpStorm, NetBeans.

Для наиболее популярных сред разработки на сегодняшний день создано огромное количество дополнительных плагинов, помогающих упростить процесс разработки программного обеспечения. Среди наиболее популярных – плагины для работы с системами контроля версий, такими как git, svn, mercurial, или, например, плагины, расширяющие поддержку языков программирования.

На данный момент для разработки хорошего программного обеспечения необходима группа разработчиков, которые разрабатывают как отдельные части системы, так и всю систему в целом. Разработчики делятся на группы по профилю - кто-то разрабатывает отдельные модули, кто-то разрабатывает интерфейс и т.п.

При этом каждый разработчик имеет дело с локальной копией проекта, а сам проект хранится на выделенном сервере с использованием системы контроля версий (VCS).

Для того, что бы иметь возможность запуска написанного приложения,

разработчикам необходимо иметь на своих компьютерах весь набор необходимого программного обеспечения - компиляторы, интерпретаторы, веб серверы, фреймворки, утилиты для работы с системами контроля версий и т.п.

С увеличением количества разрабатываемых проектов растет и количество программного обеспечения, необходимого каждому разработчику - соответственно растет и стоимость проекта, связанная со стоимостью необходимого ПО, а так же и мощность необходимого аппаратного обеспечения.

Так как, фактически, каждый разработчик работает со своей локальной копией проекта, а не непосредственно с ним на прямую, то, со стороны других разработчиков невозможно оперативно отследить вносимые в проект изменения до того момента, как разработчик разместит новую версию на сервере, а со стороны менеджмента нет возможности следить за тем, чем в данный момент занят каждый работник.

# 2. Описание решаемой проблемы и обоснование её актуальности

Одной из проблем при разработке программного обеспечения является взаимодействие разработчиков.

У современных сред разработки есть ряд существенных недостатков, с точки зрения командной разработки:

1. Отсутствие встроенных механизмов общения разработчиков - необходимо использовать стороннее программное обеспечение для общения
2. Отсутствие встроенных механизмов мониторинга состояния проекта - необходимо использовать систему контроля версий
3. В основном работа происходит не с проектом напрямую, а с его локальной копией
4. Нет возможности напрямую узнать, кто над чем работает в данный момент - внесенные в проект изменения видны только после обновления проекта в системе контроля версий
5. Всем разработчикам необходимо иметь на рабочем месте все дополнительное программное обеспечение, необходимое для разработки проекта - необходимость установки всех фреймворков, библиотек, утилит и т.п., необходимого как для функционирования IDE, так и для разработки проекта
6. Сложность со сменой рабочего места - в связи с большим количеством дополнительного ПО
7. Организация прав доступа для различных групп разработчиков - усложняется с увеличением дополнительного ПО, используемого при разработке

Данные недостатки влияют как на организацию процесса разработки, так и на скорость разработки.

Встроенные в IDE чат для общения и система контроля версий позволили бы избавиться от дополнительной установки двух серверов и множества программ на рабочие станции. При этом их использование стало бы удобнее, чем по отдельности.

Работа непосредственно с проектом, а не с его локальной копией позволила бы оперативно отслеживать изменения, вносимые в проект.

При разработке сайта на рабочем месте программиста должен быть установлен большой набор программного обеспечения - веб-сервер, СУБД, интерпретатор языка программирования, используемые при разработке фреймворки, библиотеки, плагины для библиотек и другое. Среда, позволяющая вести разработку проекта без этого набора программ позволила бы сэкономить время на настройку ПО на рабочих местах всех разработчиков, а так же на аппаратном обеспечении их рабочих мест.

# 3. Системный анализ проблемы

Цель дипломного проектирования - проектирование и разработка сервиса параллельной разработки веб-приложений, обеспечивающего уменьшение количества программного обеспечения, необходимого для ведения разработки веб-приложений, улучшение процесса разработки приложений и общения разработчиков между собой.

Критерии достижения заданной цели:

1. Уменьшение программного обеспечения, необходимого для ведения разработки
2. Встроенная в сервис система общения разработчиков
3. Обеспечение менеджмента проекта актуальными сведениями о том, чем занят конкретный разработчик

# 4. Анализ существующих решений, аналогов и прототипов

На данный момент имеется множество сред разработки, имеющих те или иные возможности по созданию, отладке и развертыванию приложений:

Cloud9 – онлайн-платформа для разработки приложений на JavaScript. На данный момент она ориентирована, скорее на единичных разработчиков, чем на команды. Обладает широкими возможностями для разработки на JavaScript, но отсутствует поддержка других языков. Поддержки управления проектами нет, оперирование идет файлами и каталогами.

Из основных достоинств можно выделить:

1. Текстовый редактор с поддержкой синтаксиса
2. Запуск и отладка кода
3. Наличие стека вызова методов
4. Обозреватель переменных

Недостатками данной системы являются:

1. Используемый в среде текстовый редактор Skywriter находится в бета-тестирования, как и сама среда Cloud9
2. Поддержка разработки только на JavaScript
3. Нет управления проектами
4. Текстовый редактор использует html5 и css3, которые на данный момент поддерживаются не всеми браузерами в полной мере (Opera, Internet Explorer)
5. Серверная часть написана так же на JavaScript, который является интерпретируемым и, как следствие, работает медленнее компилируемых языков (C, C++, C#, Java)

Хотя в планах разработчиков и есть упоминание о поддержке большего числа браузеров, но это не первоочередная задача.

Проект распространяется с открытым исходным кодом и совершенно бесплатен.

Данная система является единственной с такими условиями распространения и её единственную можно установить на внутренние сервера компании, что существенно сэкономит расходы компании, связанные с покупкой среды разработки, или оплатой аккаунтов в других системах.

Compilr – онлайн среда разработки приложений на различных языках. Поддерживаются такие языки программирования, как C++, C#, PHP, Ruby и т.д. Имеется поддержка проектов с возможностью компилирования, но отсутствует управление ими. Проект можно «расшарить» для других разработчиков, но вносимые ими изменения «не видны» до обновления страницы. Так же отсутствует взаимодействие между разработчиками.

Среди достоинств можно выделить:

1. Поддержку множества языков программирования
2. Не используется html5, что делает среду более кроссбраузерной
3. Есть подсветка синтаксиса для всех поддерживаемых языков

Недостатки:

1. Нет отладки приложений
2. Построение проекта возможно, но для запуска необходимо скачать результат компиляции
3. Закрытая лицензия – нет возможности самостоятельной установки сервиса, например, для внутреннего использования в компании
4. Нет поддержки контроля версий

Данный сервис обладает большим количеством возможностей в сравнении с предыдущим. Позволяет работать с огромным набором языков

Отсутствие системы контроля версий осложняет разработку больших проектов группами разработчиков.

Построение даже небольших проектов, а так же сохранение измененных файлов происходит заметное время, что в дальнейшем плохо скажется на разработке крупных проектов.

Проект находится не под открытой лицензией, как, например, Cloud9, что делает невозможным его установку на внутренние сервера компании. В связи с этим невозможно и хоть как-то расширить или изменить функционал системы – например, невозможно настроить компилятор / интерпретатор для запуска проекта, или, например, подключить модуль к веб-серверу.

Существенным ограничением является стоимость – бесплатный аккаунт подразумевает только создание 3 открытых проектов и 250мб места на сервере, а платные – от $4 в месяц за 25 проектов и 1Гб места до $34 в месяц за 250 проектов и 10Гб места на диске.

Coderun - так же веб-ориентированная среда разработки. Поддерживает проекты на языках С#, JavaScript, php, html, css. Так же, как и compilr не использует html5, что позволяет ей работать в большинстве браузеров.

Для проектов на языке C# есть возможность компиляции и дебагинг на стороне сервера. Так же, только для проектов на C# реализовано кодо-дополнение. В проектах на других языках на данный момент реализована только подсветка синтаксиса.

Тестовый проект можно создать без регистрации. Компиляция и сохранение файлов происходят намного быстрее, чем в compilr’e.

При покупке платного аккаунта можно получить как хостинг исходного кода в облаках, так и хостинг самого приложения – всего за $10 в месяц за 256мб RAM и 5Гб места на диске. Для особо крупных проектов есть тарифные планы до $169 в месяц за 2048мб RAM и 256Гб места на диске. Размещение проектов происходит в облаке Amazon EC2 на платформе Microsoft Windows 2003 Server (32bit). На хостинге сразу доступны Microsoft .NET 3.51, ASP.NET 2.0 и 3.5, ASP.NET Ajax, PHP 5, Silverlight 3, FrontPage и Microsoft SQL Server 2005. При желании вместо MS SQL Server можно использовать базу данных, предлагаемую Amazon’ом – SimpleDB.

Весь написанный код выполняется в песочницах, где пользователь является единственным администратором, что способствует защищенности кода.

Разрабатываемый проект можно расшарить для других разработчиков. При этом проекту присваивается уникальный URL, который можно отправлять другим разработчикам. По данной ссылке они получат отдельную, модифицируемую и полностью рабочую версию текущего кода.

Для зарегистрированных пользователей владелец общего проекта может приглашать других пользователей в проект и контролировать их права доступа.

NetBeans – офлайн среда разработки. Благодаря этому имеет практически неограниченные возможности использования языков программирования – в стандартной поставке поддерживает разработку на PHP, JavaScript, html, Java, Ruby, C/C++. Существует несколько сборок среды, поддерживающих различные технологии:

1. JavaSE – поддержка разработки на платформе JavaSE
2. JavaFX – поддержка разработки на платформах JavaSE и JavaFX
3. Java – поддержка разработки на платформах JavaSE, JavaFX, Groovy, Java Web, Java ME, Java EE
4. C/C++ – поддержка разработки на C/C++
5. PHP – поддержка разработки на PHP
6. All – поддерживает разработку на всех вышеперечисленных платформах

Можно установить любую из сборок NetBeans и в дальнейшем добавить или удалить поддержку различных технологий/языков с помощью менеджера плагинов среды.

Для установки и запуска NetBeans необходим JDK6, который можно отдельно скачать с официального сайта, или скачать набор JDK вместе со средой NetBeans для технологий Java.

Сборки NetBeans доступны для сообщества по двойной лицензии, состоящей из Common Development and Distribution License v1.0 и GNU General Public License v2.

Исходный код и сборки NetBeans так же доступны в виде архивов zip. На сайте присутствуют инструкции по установке и построению среды из исходного кода.

Среда доступна для следующих платформ:

1. Windows
2. Linux
3. Solaris
4. Mac OS X

Для среды разработано огромное количество плагинов. Они позволяют увеличить функциональность, добавляя поддержку разработки с использованием новых технологий, языков программирования. Одни плагины изменяют пользовательский интерфейс, другие – позволяют чертить UML-диаграммы или использовать системой контроля версий. Большинство плагинов выпущены так же под открытыми лицензиями и доступны бесплатно.

NetBeans – проект с открытым исходным кодом, в котором приветствуется участие разработчиков. При желании можно присоединить к проекту, стать участником сообщества и помогать проекту развиваться.

Visual Studio – интегрированная среда,упрощающая создание, отладку и развертывание прииложений. Как и NetBeans, является офлайн средой разработки.

В отличие от NetBeans Visual Studio работает только на компьютерах с операционной системой Microsoft Windows:

1. Windows XP(x86) с пакетом обновления 3 (SP3) – все выпуски, кроме Starter
2. Windows Vista(х86 и х64) с пакетом обновления 1 (SP1) – все выпуски, кроме Starter
3. Windows 7(x86 и x64)
4. Windows Server 2003(х86 и х64) с пакетом обновления 2 (SP2)
5. Windows Server 2003 R2 (х86 и х64)
6. Windows Server 2008(х86 и х64) с пакетом обновления 2 (SP2)
7. Windows Server 2008 R2(x64)

Данная особенность не позволяет использовать Visual Studio на других платформах, например Linux или Mac OS, что является серьезным недостатком.

Так же, как и NetBeans, Visual Studio существует несколько сборок, но, в отличие от NetBeans, они различаются не только поддержкой различных технологий и языков программирования, а еще и функциональностью самой среды:

1. Visual Studio Express – бесплатный выпуск. Обладает минимальными возможностями.
2. Visual Studio Professional – среда для независимых разработчиков позволяющая решать основные задачи разработки. Включает встроенную поддержку модели «разработка через тестирование», а так же инструментов отладки, которые обеспечивают создание высококлассных приложений. Стоимость – от $549
3. Visual Studio Premium – универсальный набор инструментов, упрощающий разработку приложений и предназначенный для независимых разработчиков и рабочих групп. Эта система позволяет создавать масштабируемые и высококачественные приложения. Стоимость – от $5469
4. Visual Studio Ultimate – обширный набор средств управления циклом жизни приложения для обеспечения качества от этапа проектирования до этапа развертывания. При создании новых решений или совершенствовании существующих приложений Visual Studio Ultimate позволяет воплощать идеи в жизнь благодаря поддержке различных платформ и технологий, включая облачные и параллельные вычисления. Стоимость – от $11899

Team Foundation Server (TFS) – это платформа совместной работы на основе решения Microsoft по управлению жизненным циклом приложений. TFS автоматизирует и упрощает разработку программного обеспечения, а так же обеспечивает возможность отслеживать все элементы проекта и видеть в реальном времени его состояние для всех участников проекта. Платформа так же предоставляет мощные средства панели мониторинга и создания отчетов.

Выделим наиболее значимые характеристики сред разработки, необходимые для ведения параллельной разработки веб-проектов, в сводную таблицу:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | cloud9 | compilr | coderun | netbeans | TFS+VS |
| Поддержка VCS | - | - | - | доступна через плагины | + |
| Поддержка управления проектами | - | - | - | - | + |
| Поддержка языков программирования | JavaScript | C, C++, C#, PHP, Ruby, JavaScript, VB, Java | C#, PHP, Javascript, HTML, CSS | C, C++, Java, PHP, CSS, JavaScript Остальные языки достпуны через плагины бесплатно. | C++, C#, F#, J#, JavaScript, VB.NET. Остальные языки доступны через плагины платно. Плагин для PHP - $99 |
| Параллельное редактирование 1 файла | - | - | - | - | - |
| Запуск проекта | + | только компиляция, для запуска нужно скачать проект | + | + | + |
| Встроенная система общения | - | - | - | - | - |
| Цена | бесплатно, открытый исходный код | тестовый проект - бесплатно, оплачиваемые аккаунты - от $4 до $34 в месяц | тестовый проект - бесплатно, оплачиваемые аккаунты - от $10 до $169 в месяц | бесплатно, открытый исходный код | TFS - $499, VS - от бесплатной лицензии до $11899, VS.php - $99 |

Как видно из приведенной таблицы – даже дорогостоящие решения не могут полностью удовлетворить заявленным требованиям, а бесплатные системы не удовлетворяют даже половине поставленных требований.

Ни одна из выше перечисленных сред разработки приложений не поддерживает работу с системой управления версиями по умолчанию.

Система управления версиями (Version Control System, VCS) – программное обеспечение для облегчения взаимодействия 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.

Ситуация, когда документ за время своего существования претерпевает ряд изменений, достаточно типична. В простейшем случае можно хранить несколько вариантов документа, соответствующим образом их нумеруя. Но такой способ крайне не эффективен (приходится хранить несколько практически идентичных копий), требует повышенного внимания и дисциплины и часто ведет к ошибкам. По этому были разработаны средства для оптимизации этой работы.

Традиционные системы управления версиями используют традиционную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельта-компрессию — такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Иногда создание новой версии выполняется незаметно для пользователя (прозрачно), либо прикладной программой, имеющей встроенную поддержку такой функции, либо за счёт использования специальной файловой системы. В этом случае пользователь просто работает с файлом, как обычно, и при сохранении файла автоматически создаётся новая версия.

Часто бывает, что над одним проектом одновременно работают несколько человек.

Некоторые системы управления версиями дают возможность заблокировать файл в хранилище. Блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла (например, средствами файловой системы) и обеспечивает, таким образом, исключительный доступ только тому пользователю, который работает с документом.

Многие системы управления версиями предоставляют ряд других возможностей:

1. Позволяют создавать разные варианты одного документа, т. н. ветки, с общей историей изменений до точки ветвления и с разными — после неё.
2. Дают возможность узнать, кто и когда добавил или изменил конкретный набор строк в файле.
3. Ведут журнал изменений, в который пользователи могут записывать пояснения о том, что и почему они изменили в данной версии.
4. Контролируют права доступа пользователей, разрешая или запрещая чтение или изменение данных, в зависимости от того, кто запрашивает это действие.

Распределённые системы управления версиями – также известны как Distributed Version Control System, DVCS. Такие системы используют распределённую модель вместо традиционной клиент-серверной. Они, в общем случае, не нуждаются в централизованном хранилище: вся история изменения документов хранится на каждом компьютере, в локальном хранилище, и при необходимости отдельные фрагменты истории локального хранилища синхронизируются с аналогичным хранилищем на другом компьютере. В некоторых таких системах локальное хранилище располагается непосредственно в каталогах рабочей копии.

Когда пользователь такой системы выполняет обычные действия, такие как извлечение определённой версии документа, создание новой версии и тому подобное, он работает со своей локальной копией хранилища. По мере внесения изменений, хранилища, принадлежащие разным разработчикам, начинают различаться, и возникает необходимость в их синхронизации. Такая синхронизация может осуществляться с помощью обмена патчами или так называемыми наборами изменений между пользователями.

Описанная модель аналогична созданию отдельной ветки для каждого разработчика в классической системе управления версиями (в некоторых распределённых системах перед началом работы с локальным хранилищем нужно создать новую ветвь). Пока разработчик изменяет только свою ветвь, его работа не влияет на других участников проекта и наоборот. По завершении обособленной части работы, внесённые в ветви изменения сливают с основной (общей) ветвью. Как при слиянии таких ветвей, так и при синхронизации разных хранилищ, возможны конфликты версий. На этот случай во всех системах предусмотрены те или иные методы обнаружения и разрешения конфликтов слияния.

Основное преимущество распределённых систем заключается в их гибкости.

Каждый разработчик может вести работу независимо, так, как ему удобно, сохраняя промежуточные варианты документов и передавая результаты другим участникам, когда посчитает нужным. При этом обмен наборами изменений может осуществляться по различным схемам. В небольших коллективах участники работы могут обмениваться изменениями по принципу «каждый с каждым», за счет чего отпадает необходимость в создании выделенного сервера. Крупное сообщество, наоборот, может использовать централизованный сервер, с которым синхронизируются копии всех его участников. Возможны и более сложные варианты — например, с созданием групп для работы по отдельным направлениям внутри более крупного проекта.

Так как современные среды разработки без дополнительных плагинов не поддерживают работу с системами контроля версий, то необходимо провести сравнительный анализ существующих систем для выбора наиболее подходящей для использования в разрабатываемой системе.

Git (гит) – распределенная система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux. Программа является свободной и выпущена под лицензией GNU GPL версии 2.

Система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы.

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-daemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использование существующих конфигураций сетевых фильтров.

Mercurial (англ. Ртутный) – кроссплатформенная распределенная система управления версиями, разработанная для эффективной работы с большими репозиториями кода. В первую очередь является консольной программой. Все операции запускаяются параметрами команды hg, название которой взято от обозначения химического знака ртути.

Система Mercurial написана на Python, хотя чувствительные к производительности части (например, своя реализация diff) выполнены в качестве Python-расширений на C. Репозитории Mercurial управляются при помощи утилиты командной строки hg.

Наряду с традиционными возможностями систем контроля версий, Mercurial поддерживает полностью децентрализованную работу (отсутствует понятие основного хранилища кода), ветвление (возможно вести несколько веток одного проекта и копировать изменения между ветками), слияние репозиториев (чем и достигается «распределённость» работы). Поддерживается обмен данными между репозиториями через HTTP/HTTPS, SSH и вручную при помощи упакованных наборов изменений.

Mercurial использует SHA1-хеши для идентификации ревизий и позволяет присваивать отдельным ревизиям индивидуальные метки.

Утилита hg обладает компактным интерфейсом, и Mercurial считается более простой в освоении системой, чем, например, git.

В комплекте к Mercurial поставляются CGI-сценарии для предоставления web-интерфейса к репозиториям.

Bazaar (ранее известная как Bazaar-NG, имя утилиты командной строки bzr) — распределённая система управления версиями, разработка которой спонсируется фирмой Canonical Ltd.. Система Bazaar разработана с целью облегчить работу над развитием свободных и открытых проектов для всех желающих.

Команда разработчиков фокусируется на том, чтобы сделать систему лёгкой в использовании, но при этом точной в деталях и очень гибкой, подстраиваемой под конкретные нужды пользователей. Также большое внимание уделяется вопросам работы со множеством веток и их последующим объединением. Bazaar может использоваться как одним разработчиком для работы над множеством локальных веток, так и группой разработчиков, совместно работающих в сети.

Система контроля версий Bazaar написана на языке программирования Python. Существуют установочные пакеты для основных дистрибутивов Linux, инсталляторы для Mac OS X и MS Windows. Bazaar — это свободное программное обеспечение, в настоящее время является частью проекта GNU

В отличие от чисто распределённых систем контроля версий, которые не используют центральный сервер, Bazaar поддерживает работу как с сервером так и без него. Возможно даже использовать оба метода одновременно для одного и того же проекта.

Bazaar поддерживает работу напрямую с некоторыми другими системами контроля версий. Пользователи могут создавать новые ветки на основе репозиториев других систем (таких как Subversion или Git), делать локальные изменения и фиксировать их в Bazaar-ветке, и затем отправлять свои изменения назад в оригинальный репозиторий.

Subversion — централизованная система управления версиями (в отличие от распределённых систем, таких как Git или Mercurial), то есть данные хранятся в едином хранилище. Хранилище может располагаться на локальном диске или на сетевом сервере.

Работа в Subversion мало отличается от работы в других централизованных системах управления версиями. Клиенты копируют файлы из хранилища, создавая локальные рабочие копии, затем вносят изменения в рабочие копии и фиксируют эти изменения в хранилище. Несколько клиентов могут одновременно обращаться к хранилищу. Для совместной работы над файлами в Subversion преимущественно используется модель Копирование — Изменение — Слияние. Кроме того, для файлов, не допускающих слияние (различные бинарные форматы файлов), можно использовать модель Блокирование — Изменение — Разблокирование.

При сохранении новых версий используется дельта-компрессия: система находит отличия новой версии от предыдущей и записывает только их, избегая дублирования данных.

При использовании доступа с помощью WebDAV также поддерживается прозрачное управление версиями — если любой клиент WebDAV открывает для записи и затем сохраняет файл, хранящийся на сетевом ресурсе, то автоматически создаётся новая версия.

Subversion предоставляет следующие способы доступа к репозиторию:

1. прямой доступ к репозиторию на диске (на локальной или сетевой файловой системе)
2. удалённый доступ по протоколу WebDAV/DeltaV поверх HTTP (или HTTPS) с использованием модуля mod\_dav\_svn для веб-сервера Apache 2
3. удалённый доступ с использованием собственного протокола SVN:
4. на выделенном сетевом соединении (по умолчанию на TCP-порту 3690)
5. через стандартный ввод-вывод (в том числе через средства удаленного CLI, например SSH)

Все эти способы могут быть использованы для работы с репозиториями обоих типов (FSFS и Berkeley DB). Для доступа к одному и тому же репозиторию могут одновременно использоваться разные способы.

# 5. Постановка общей задачи проектирования и разработки сервиса

Задача проектирования – создать среду разработки веб-приложений, позволяющую разработчикам удаленно работать над одним и тем же проектом без настройки и установки дополнительного программного обеспечения.

Наиболее часто при разработке крупных проектов используется VCS и один из механизмов общения – чаты, skype-конференции, группы в социальных сетях.

Основной функцией разрабатываемой системы является обеспечение комфортной работы группы разработчиков над веб-проектом

В связи с этим можно определить следующие требования к разрабатываемой системе:

1. Поддержка управления версиями
2. Чат для общения между разработчиками
3. Механизмы мониторинга действий разработчиков
4. Механизмы для управления правами различных групп разработчиков
5. Добавление / удаление разработчиков проекта
6. Механизмы запуска и отладки проекта
7. Редактор кода с подсветкой синтаксиса и отображением действий других разработчиков

К интерфейсу разрабатываемой среды можно определить следующие требования:

1. Интерфейс должен внешне быть похож на интерфейсы наиболее популярных на сегодняшний день сред разработки (Microsoft Visual Studio, NetBeans, Intellij Idea)
2. Возможность пользовательских настроек некоторых параметров редактора кода (шрифт, размер шрифта)
3. Возможность «свернуть» некоторые элементы среды (например, чат) в пиктограмму, для увеличения рабочего пространства редактора кода.

# 6. Перечень задач, подлежащих решению в процессе разработки

Так как архитектура приложения клиент-серверная, то задачи, подлежащие решению в процессе разработки можно разделить на 3 группы:

1. Задачи, относящиеся к клиентской части
2. Задачи, относящиеся к серверной части
3. Задачи, относящиеся к взаимодействию между клиентской и серверной частью.

Задачи, относящиеся к клиентской части в основном связаны с взаимодействием с пользователем:

1. Проектирование и разработка пользовательского интерфейса компонентов среды
2. Проектирование и разработка форм, диалогов общения системы с пользователем
3. Проектирование и разработка меню
4. Определение палитры приложения

Задачи, относящиеся к взаимодействию между серверной и клиентской частью:

1. Определения механизма обмена информацией между клиентом и сервером
2. Определение протокола взаимодействия клиента с сервером

Задачи, относящиеся к серверной части связаны с организацией внутренней работы сервиса:

1. Проектирование структуры базы данных
2. Проектирование внутренних и внешних структур данных
3. Проектирование архитектуры приложения
4. Проектирование интерфейсов подключаемых модулей
5. Организация работы с базой данных
6. Организация работы с проектами пользователей