**ВВЕДЕНИЕ**

С приходом более совершенных IDE и языков программирования, вопрос взаимодействия с коллегами является ключевым для многих разработчиков. В основном, чтобы решить эту проблему, начали появляться централизованные системы контроля версий (ЦСКВ или CVCs – Centralized Version Control systems).

В них используется следующий метод сохранения информации: имеется один сервер, который хранит все версии файлов, а также некоторое количество клиентов, на которые поставляется информация с сервера. Такой подход имеет много преимуществ, особенно относительно RCS – Revision Control System, системы, которая является одной из самых первых систем управления версиями.

Для каждого файла, зарегистрированного в системе, ЦСКВ хранит полную историю изменений, причём для текстовых файлов используется эффективный алгоритм дельта-компрессии, когда хранится только последняя версия и все межверсионные изменения. Система позволяет также хранить версии бинарных файлов, но без использования этого механизма, то есть каждая версия бинарного файла хранится полностью.

Но, несмотря на решение достаточно большого спектра проблем, у таких систем есть свои недостатки. Первый и наиболее существенный – единая точка отказа. Если сервер по каким-то причинам выходит из строя, то все работают только со своими локальными копиями, не имея возможности смотреть изменения других разработчиков.

Чтобы решить эту проблему и забыть о проблемах RCS, существуют децентрализованные системы контроля версий (ДСКВ или DVCS – Distributed Version Control System).

Как следует из названия, в таких системах клиенты не просто забирают последнюю (или выбранную, но всегда одну) версию проекта: на самом деле они полностью копируют его содержание и историю. Таким образом, при таком подходе у системы появляются два больших преимущества перед CVCs:

− разработчик может посмотреть состояние проекта в любой момент времени в прошлом, даже без наличия доступа к центральному серверу;

− каждая копия проекта это, по сути дела, полная резервная копия проекта. Таким образом, если сервер выйдет из строя, то любой разработчик сможет восстановить его состояние на любой другой машине.

Наиболее популярными системами контроля версий являются Mercurial, SVN и Git системы контроля версий.

На предприятиях Республики Беларусь используется достаточно много различных систем контроля версий (VCS). Это объясняется спецификой заказов, а именно необходимостью поддерживать различные устаревшие проекты. Тем не менее, основную часть VCS составляют системы Mercurial и Git. На них пишутся большинство новых приложений и проектов. Во время прохождения практики мною были изучены возможности именно Git-систем в рамках сразу пяти проектов. Таким образом, можно сделать вывод, что именно Git-системы являются преобладающими.

Тем не менее, также большое количество проектов написаны с использованием SVN-систем. В основном это достаточно долгосрочные проекты для крупных международных организаций, то есть те, старт которых был дан примерно в прошлом десятилетии.

Основная причина использования системы Git заключается в её популярности. Система Git является одним из основных инструментов, которых требуют заказчики, а кроме того, именно для работы с Git написана самая популярная система для работы с файлами – GitHub. Её популярность, в свою очередь, обусловлена достаточно большим набором возможностей и команд:

* поддержка меток;
* функция сохранения временного состояния (англ. stash);
* качественный веб-интерфейс «из коробки»;
* продуманная система команд, позволяющая удобно встраивать функции Git в скрипты.

Основная причина использования системы Mercurial состоит в том, что она достаточно легка для понимания и изучения, особенно для разработчиков, для которых первой системой контроля версий была SVN и подобные. Также система Mercurial исторически лучше поддерживает операционную систему Windows, в которой у начинающего разработчика будет меньше проблем с настройкой.

Основные причины использования системы Subversion – наследие и legacy-код. Однако, существуют и плюсы SVN относительно Git:

* отслеживает версии не только файлов, но и директорий;
* атомарность многофайловых фиксаций;
* поддерживаются наборы изменений (англ. changeset);
* с каждым файлом и директорией может быть связан произвольный набор свойств, состоящих из названия и значения. Свойства тоже находятся под управлением версиями;
* одинаково эффективно работает как с текстовыми, так и с двоичными файлами.

В данном дипломном проекте для изучения систем контроля версий будут предложены Git как децентрализованная система контроля версий, система SVN (Subversion) как централизованная и система Mercurial в качестве промежуточного звена.