**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 (ЦСВ или CVS – Centralized Version Control systems). Примерами таких систем являются:

* CVS;
* Subversion;
* Perforce.

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. Диаграмма таких взаимоотношений представлена на схеме 1:



Схема 2.1 – Диаграмма взаимоотношений в VCS

Такой подход имеет много преимуществ, особенно относительно RCS. Основное из них в том, что разработчики могут работать с одним и тем же файлом одновременно. Кроме этого, есть ещё и другие плюсы, например, каждый разработчик в любой момент времени может знать, что делают все остальные; кроме этого, администраторы систем, имея точно такую же возможность, могут эффективнее управлять своими кадрами и всем продуктом в целом.

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, в свою очередь забыть о проблемах RCC, существуют децентрализованные системы контроля версий (ДСКВ или DVCS – Distributed Version Control System). В качестве примеров можно привести следующие ДСКВ:

* Git;
* Mercurial;
* Darcs;
* Bazaar.

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 VCS:

* В любой момент времени разработчик может посмотреть состояние проекта в какой-то момент времени 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.

Диаграмма взаимоотношений между клиентом и сервером представлена на схеме 2:



Схема 2.2 – Децентрализованная система контроля версий

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

Тем ни 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 дал примерно в прошлом десятилетии.

На основании полученных данных, в моей дипломной работе будут использоваться git как децентрализованная система и svn (subversion) как централизованная.

Основная причина в использовании git заключается в его популярности. Git является одним из основных инструментов, которые требуют заказчики, а также в том, что именно для работы с git написан самая популярная работа для работы с файлами – GitHub.

В данном дипломном проекте описаны методы тестирования, рассмотрены модели жизненного цикла программного продукта, приведены метрика качества, приведены примеры автоматизированного тестирования, составлены тестовые задания для проведения лабораторного практикума по тестированию программного кода.