DOI: 10.17117/na.2015.12.02.022

<http://ucom.ru/doc/na.2015.12.02.022.pdf>

Поступила (Received): 31.12.2015

Архипов Л.В. Концепция ветвления процесса разработки проекта в системах контроля версий

Arkhipov L.V. The concept of version control systems for branching of the software development

В статье проводится обзор концепции ветвления процесса разработки проекта программного обеспечения в системах контроля версий на примере Git и делается вывод о целесообразности применения данной концепции для процесса разработки проекта ПО

Ключевые слова: версия, фиксация, ветвь

The article provides an overview of concept of version control system for branching software development on the example of Git. The conclusion of the feasibility of implementing these systems

Key words: version, commit, branch

Архипов Леонид Владиславович

Аспирант

Пензенский государственный университет

г. Пенза, ул. Красная, 40

Arkhipov Leonid Vladislavovich

Graduate

Penza state university

Penza, Krasnaya st., 40

На сегодняшний день обеспечение качества программного продукта (ПП) любого уровня, от простого приложения смартфона до автоматизированной системы, обслуживающей нужды страны, – важная задача для разработчиков, решение которой – использование эффективных методов организации качественного процесса разработки ПП в условиях ограниченности ресурсов и сроков. Основной метод обеспечения качества процесса разработки – выбор и следование стандартам, моделям и методологиям разработки ПО, парадигмам программирования, формирующим необходимые требования к качеству, модель жизненного цикла ПП, структуру процесса исходя из специфики проекта, ограничений по срокам и ресурсам. Данный метод предусматривает обязательный контроль качества: проведение разнообразных видов тестирования, верификации, вычисление метрик. Дополнительный метод – использование современных инструментов упрощающих процесс разработки. Разнообразные программные реализации решают задачи автоматизации рутинного труда программиста, организации коллективного рабочего процесса, тестирования и отладки исходного кода. Обязательный инструмент организации процесса разработки – система контроля версий (СКВ). СКВ – это программное обеспечение, отслеживающее изменения в одном или нескольких файлах, с возможностью сохранения и вос-

The diagram illustrates the workflow of a distributed version control system (VCS) using a series of nodes and directed edges representing operations.

Nodes (Commits):

- Фиксация 1** (Green circle): readme.txt
- Фиксация 2** (Yellow circle): readme.txt, file.txt [1kb]
- Фиксация 2.1** (Yellow circle): readme.txt, file.txt [1kb], test.png
- Фиксация 1.1** (Green circle): readme.txt, file.txt [5kb]
- Фиксация 3** (Yellow circle): readme.txt, file.txt [7kb], test.png
- Фиксация 3.1** (Pink circle): readme.txt, file.txt [7kb]
- Фиксация 3.2** (Pink circle): images/test.png, file.txt [7kb], readme.txt
- Фиксация 1.2** (Green circle): readme.txt, file.txt [10kb]
- Фиксация 1.3** (Green circle): readme.txt, file.txt [11kb]

Operations (Edges):

- Фиксация 1 → Фиксация 2:** добавление file.txt [1kb] (Brown arrow)
- Фиксация 2 → Фиксация 2.1:** добавление test.png (Brown arrow)
- Фиксация 2.1 → Фиксация 3:** удаление test.png (Brown arrow)
- Фиксация 1 → Фиксация 1.1:** добавление file.txt [5kb] (Green arrow)
- Фиксация 1.1 → Фиксация 3:** слияние ветви 2 и ветви 3 (Green arrow)
- Фиксация 1.1 → Фиксация 1.2:** изменение file.txt (Green arrow)
- Фиксация 3 → Фиксация 3.1:** добавление test.png в images (Pink arrow)
- Фиксация 3.1 → Фиксация 3.2:** добавление test.png в images (Pink arrow)
- Фиксация 1.2 → Фиксация 1.3:** изменение file.txt (Green arrow)

Branching and Merging:

- HEAD** points to **Ветвь 3** (Branch 3), which points to **Фиксация 3.2**.
- Ветвь 2** (Branch 2) points to **Фиксация 3**.
- Ветвь 1** (Branch 1) points to **Фиксация 1.3**.

Ветвь в Git – указатель на фиксацию. Перемещение ветви по графу фиксаций удаляет текущее содержимое репозитория, и восстанавливает версию проекта из фиксации, на которую переместился указатель ветви. Указатель HEAD сообщает Git текущую ветвь, фиксация которой задаёт состояние репозитория. При переключении HEAD на другую ветвь Git переводит репозиторий в состояние фиксации, на которую соответствующая ветвь. При внесении в последнюю версию проекта изменений, и последующем создании новой фиксации, Git перемещает указатели HEAD и текущей ветви на созданную фиксацию. Каждая фиксация может иметь неограниченное количество потомков, создаваемых под разными ветвями, но максимальное количество предков – 2, достигается лишь в результате операции слияния. При выполнении операции слияния ветвей Git сравнивает содержимое соответствующих фиксаций, затем объединяет состояния репозитория разных версий проекта, формируя новую версию: копируются

недостающие файлы, в существующие файлы дописываются данные из разных версий. На основе новой версии создаётся фиксация, на которую Git переводит указатели HEAD и ветвей, участвовавших в слиянии. Если на этапе сравнения обнаружены изменения в одном и том же бинарном файле, или в одних и тех же строках текстового файла, Git сообщает о конфликте содержимого, который должен разрешить пользователь, выбрав изменение над файлом одной из версий, либо внося собственные. Помимо информации о предках и потомках, каждая фиксация содержит данные о дате создания, авторе, и сообщении, состоящее из описания изменений, отличающих соответствующую версию проекта, и обоснования внесения данных изменений в проект. Написание поясняющего сообщения – обязательный этап при создании фиксации. При решении конкретной задачи, возникающей в процессе разработки проекта, организация ветвления и способ хранения версий Git позволяет разработчику использовать локальную копию репозитория, внося любые изменения в общий проект через собственную ветвь разработки, которая по завершению работы над задачей сливается с основной ветвью проекта.

Решающий фактор успешности разработки ПО – организация обмена необходимой информацией между разработчиками, на основе которой формируется правильное представление о проекте в целом и решаемых в процессе разработки задачах [3]. Важная часть обмена информацией – ревизии чужого исходного кода, с целью синхронизации и использования наработок других разработчиков при реализации собственных функций. При согласовании группой разработчиков унифицированного формата поясняющих сообщений к фиксациям упрощается процесс понимания чужого кода. Использование ключевых слов из списка определённого разработанным форматом в поясняющих сообщениях позволяет более эффективно осуществлять поиск по графу фиксаций, применять метрики. Список ключевых слов основывается на классификации фиксаций по типу вносимых изменений: изменения структуры проекта (добавление, удаление файлов или каталогов), изменения по расширению функционала, изменения по отладке кода, изменения по рефакторингу кода.

Разработка ПО невозможна без использования современных инструментов, упрощающих всё большее число аспектов процесса создания ПО. Однако для достижения более эффективных результатов при использовании данных инструментов требуется разработка алгоритмов по ведению рабочего процесса, на основе которых возможна дальнейшая автоматизация.

Список используемых источников:

1. Какую систему управления версиями вы используете (в реальной работе, больше всего)?
URL: <http://habrahabr.ru/post/269515/>
2. Chacon S., Straub B. Pro Git.
URL: <https://progit2.s3.amazonaws.com/en/2015-12-30-89ed8/progit-en.972.pdf>
3. Брукс Ф. Мифический человеко-месяц. Символ-Плюс, 2001. 304 с.