Департамент образования Белгородской области
Областное государственное автономное профессиональное образовательное учреждение
«Белгородский индустриальный колледж»

Методы организации работы в команде разработчиков. Системы контроля версий

Выполнил: Кощин Д.С.

Руководитель: Дьякова А.К.

Белгород 2023

Модели коллективов разработчиков

Разработки программы в зависимости от количества участников и типов взаимоотношений между участниками делятся на виды:

- **Авторская разработка**
- **Коллективная разработка**
- **Общинная модель разработки**

Авторская разработка

Авторская разработка – это создание программных продуктов, при котором весь жизненный цикл разработки поддерживается одним человеком. Был распространён в г.г. и сейчас применяется редко из-за сложности, объема, требуемого качества, сопровождения.

С появлением ПК программное обеспечение стало продуктом массового применения, поэтому стали доминировать крупные компьютерные компании с развитой структурой менеджмента и рекламной компанией. Авторские разработки применяются в области наукоемких приложений и при создании условно бесплатных программных продуктов. Для таких приложений характерна необходимость многолетнего изучения предметной области, практически полное отсутствие начального финансирования проекта, малая рентабельность, которая определяется узким кругом области.

Коллективная разработка

Коллективная разработка – это создание программного продукта коллективом разработчиков. Один из основных вопросов коллективной разработки является разделение труда. Один человек не способен создать приложение масштаба предприятия. Ни один разработчик просто не удержит в голове все требования к системе и варианты проекта. Поэтому сегодня разработкой промышленных систем занимаются проектные группы, и все обязанности распределяются среди членов группы.

Существует две основные модели организации коллектива при разработке ПО:

1) иерархическая модель (начальник -> подчинённые)

2) модель группы (не определяет структуру коллектива с точки зрения отдела кадров)

Основной моделью разделения труда в наше время является иерархическая структура.

Иерархическая модель

Если в современных производственных средах один менеджер проекта (начальник) отвечает за все тонкости разработки и принимает все важные решения, возникает множество проблем, ведущих к провалу проекта. **Иерархическая модель грешит множеством недостатков:**

- нехватка информации
- невозможность учесть все особенности проекта
- отсутствие полноценной связи между всеми участниками проекта
- трудность освоения новых технологий
- сложность расстановки приоритета
- структура равноправности соисполнителей

Кроме того, опыта одного человека чаще всего недостаточно для быстрого решения задачи и для интеграции приложения в существующую инфраструктуру.

В организациях, построенных на основе иерархической модели, затруднен обмен информацией в этой модели он, по определению, осуществляется через посредников.

Модель группы

Дабы сгладить недостатки иерархической модели, в проектной группе предусматривается распределение обязанностей руководителя между членами коллектива. При этом за проект отвечает не один человек, а все члены группы каждый за свой участок.

Модель группы не определяет структуру коллектива с точки зрения отдела кадров. Задача модели проектной группы определить цели проекта и распределить обязанности.

Руководители каждого направления с помощью выделенных им ресурсов выполняют возложенную на них часть работы. Обязанности ролей определяются работой над проектом, а не деятельностью «штатной единицы»..

И у коллективного подхода есть недостатки:

- разрозненная связь с внешними источниками информации;
- несогласованное представление о разных сторонах проекта;
- несогласованность личных планов членов группы;
- отсутствие опыта, снижающее эффективность коллективной работы.

Общинная модель разработки

Идеология общинной ("базарной") модели разработки сформулирована в программной статье Эрика Раймонда (Eric Raymond) "Собор и Базар".

Общинная модель характеризуется тремя основными факторами:

- Децентрализованностъ разработки. Не существует ограничения сверху на количество людей, принимающих участие в проекте. Как правило, разработки такого типа ведутся на базе сети Интернет и могут включать любого заинтересованного разработчика Сети.
- Разработка ведется на базе открытых исходных текстов. По ним можно разобраться с сутью задачи и в любой момент подключиться к разработке.
- Большое количество внешних тестеров (бета-тестеров), позволяющих быстро обнаруживать ошибки и проблемы в программе.

Эрик Рэймонд сформулировал несколько уроков, которые позволяют лучше понять особенности общинной разработки:

- Каждая хорошая программа начинается с энтузиазма разработчика.
- Хорошие программисты знают, что можно написать, а великие — что можно переписать.
- При правильном отношении интересная проблема найдет вас сама.
- Когда вы теряете интерес к программе, ваша последняя обязанность -передать ее компетентному преемнику.
- Следует выпускать ранние и частые версии программ.
- Обнаружить проблему и исправить ее могут разные люди.
- Иногда использовать идеи пользователей лучше, чем свои идеи.

Общинная модель разработки

Идеология общинной ("базарной") модели разработки сформулирована в программной статье Эрика Раймонда (Eric Raymond) "Собор и Базар".

Общинная модель характеризуется тремя основными факторами:

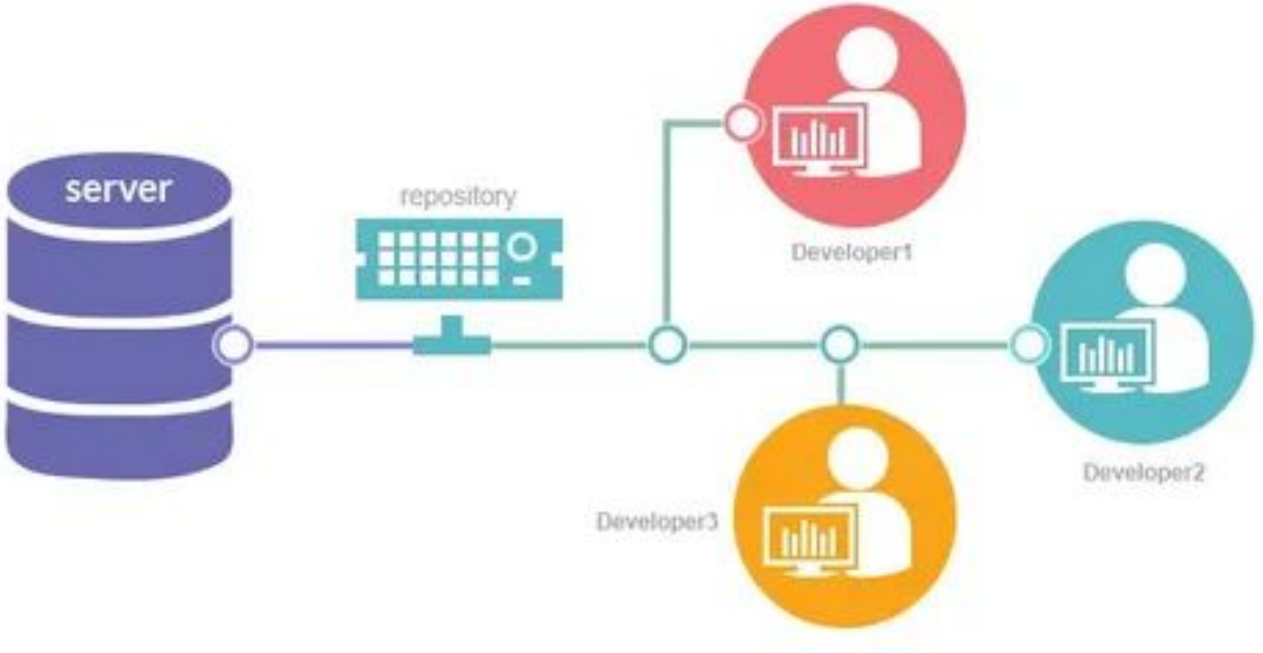
- Децентрализованностъ разработки. Не существует ограничения сверху на количество людей, принимающих участие в проекте. Как правило, разработки такого типа ведутся на базе сети Интернет и могут включать любого заинтересованного разработчика Сети.
- Разработка ведется на базе открытых исходных текстов. По ним можно разобраться с сутью задачи и в любой момент подключиться к разработке.
- Большое количество внешних тестеров (бета-тестеров), позволяющих быстро обнаруживать ошибки и проблемы в программе.

Эрик Рэймонд сформулировал несколько уроков, которые позволяют лучше понять особенности общинной разработки:

- Каждая хорошая программа начинается с энтузиазма разработчика.
- Хорошие программисты знают, что можно написать, а великие — что можно переписать.
- При правильном отношении интересная проблема найдет вас сама.
- Когда вы теряете интерес к программе, ваша последняя обязанность -передать ее компетентному преемнику.
- Следует выпускать ранние и частые версии программ.
- Обнаружить проблему и исправить ее могут разные люди.
- Иногда использовать идеи пользователей лучше, чем свои идеи.

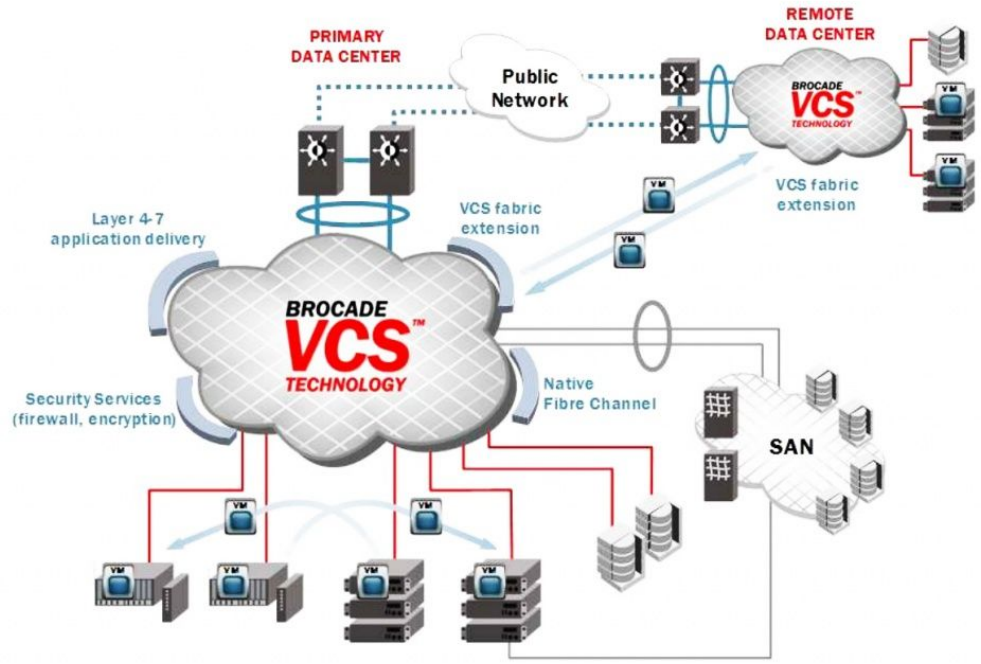
VCS

Система управления версиями (от англ. *Version Control System*, VCS или *Revision Control System*) — 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



Назначение VCS

- Архивация и восстановления
- Синхронизация работы с командой
- Поиск “виновного”
- Хранения истории разработки
- Отмена изменений
- Альтернативные/экспериментальные реализации



Классификация систем контроля версий

- Централизованные/распределённые — в централизованных системах контроля версий вся работа производится с центральным репозиторием, в распределённых — у каждого разработчика есть локальная копия репозитория.

- Блокирующие/не блокирующие — блокирующие системы контроля версий позволяют наложить запрет на изменение файла, пока один из разработчиков работает над ним, в неблокирующих один файл может одновременно изменяться несколькими разработчиками.

- Для текстовых данных/для бинарных данных — для VCS для текстовых данных очень важна поддержка слияния изменений, для VCS с бинарными данными важна возможность блокировки.

Ветвление

- Ветвь (branch) — направление разработки проекта, независимое от других.
- Ветвь представляет собой копию части (как правило, одного каталога) хранилища, в которую можно вносить свои изменения, не влияющие на другие ветви.
- Документы в разных ветвях имеют одинаковую историю до точки ветвления и разные — после неё.
- Изменения из одной ветви можно переносить в другую.
- Ствол (trunk, mainline, master) — основная ветвь разработки проекта.

GIT

- **Git** — распределённая система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux в 2005 году.

Плюсы:

- Высокая производительность.
- Развитые средства интеграции с другими VCS.
- Репозитории git могут распространяться и обновляться общесистемными файловыми утилитами, такими как rsync.

Схема работы с Git

- **git pull/ fetch** — забираем изменения из центрального репозитория
- **git push** — вносим изменения в удаленный репозиторий
- **git commit** — совершение коммита
- **git checkout** — переключение между ветками, извлечение файлов
- **git merge** — слияние веток (разрешение возможных конфликтов)
- **git add** — позволяет внести в индекс— изменения, которые затем войдут в коммит

Вопросы

- **Какие есть виды разработки программы в зависимости от количества участников и типов взаимоотношений между участниками?**
- **Какие существуют основные модели организации коллектива при разработке ПО?**
- **Для чего нужна система управления версиями?**
- **Какие существуют классификации систем контроля версий?**
- **Что такое ветвь?**