Государственное автономное профессиональное образовательное учреждение Пензенской области «Пензенский колледж информационных и промышленных технологий (ИТ-колледж)»

**Лабораторная работа № 7 по дисциплине**«Внедрение и поддержка».

на тему: «Конфигурирование программных и аппаратных средств»

Выполнил:

Крючков Евгений

ст.гр. 22ИТ35

Проверил преподаватель:

Николаева Н.С.

Пенза, 2024

**Цель работы:** получение навыков произведения настройки и конфигурирования ПО КС.

**Материалы, оборудование, программное обеспечение:** персональный компьютер, операционная система Windows, методические рекомендации.

**Ход работы**

План управления конфигурацией ПО для приложения "калькулятор RomuL" младшим классам в образовательные учреждения.

1. **Цель:** цель приложения состоит в привлечении молодого поколения к изучению математики путём игр.
2. **Объем:** План охватывает все компоненты программного обеспечения "калькулятор RomuL!", включая исходный код, исполняемые файлы, документацию (техническую, пользовательскую), тестовые данные, конфигурационные файлы и скрипты.
3. **Методология:** Используется методология управления версиями Git с использованием GitFlow или аналогичного ветвления.
4. **Инструменты:**

* Система контроля версий: GitHub.
* Система управления задачами: Trello;

1. **Процессы:**

* Управление версиями: Все компоненты ПО хранятся в репозитории Git. Используется семантическое версионирование (Semantic Versioning - SemVer).
* Ветвление (branching): Используется стратегия ветвления, соответствующая выбранной методологии (GitFlow: master, develop, feature branches, release branches, hotfix branches).
* Слияние (merging): Все изменения проходят код-ревью перед слиянием в основную ветку (develop).
* Тестирование: После каждого изменения проводится соответствующее тестирование (юнит-тесты, интеграционные тесты, функциональные тесты).
* Релизы: Релизы создаются из стабильных release-ветвей. Создаются release notes с описанием изменений в каждой версии.
* Документация: Вся документация (техническая, пользовательская) хранится в системе контроля версий. Используется Markdown или другой удобный формат.
* Управление изменениями: Все изменения в коде и документации должны быть задокументированы в системе управления задачами.

1. **Ответственности:**

* Разработчик: Крючков Евгений.
* Тестировщик: Крючков Евгений.
* Менеджер проекта: Крючков Евгений.

1. **Метрики:**

* Количество коммитов в неделю: не менее 5.
* Количество открытых pull requests: не более 5.
* Время разрешения pull requests: не более 24 часов
* Количество найденных ошибок на этапе тестирования: не более 3 на релиз.
* Количество релизов в неделю/месяц: не менее 1.

1. **Аудит:** Регулярный аудит конфигурационного управления будет проводиться раз в квартал для проверки соответствия установленным правилам и процессам.

**PVCS (Merant)**

PVCS помогает группам разработки управлять параллельной разработкой.

Считается, что для предотвращения разрушительных технологических ошибок, таких как проблема 2000 года, деятельность разработчиков нужно контролировать. Эту задачу можно было бы возложить на некоего редактора. Сама по себе идея возражений не вызывает; однако код, который пишут программисты, имеет так много взаимосвязей с другими объектами, что проверить корректность одного модуля вовсе не значит дать гарантию от ошибок, которые могут дорого обойтись.

В качестве возможного решения можно предложить развернуть систему управления конфигурацией программного обеспечения (SCM), которая позволяет отслеживать процесс создания приложения с самого начала. Компания Merant предлагает следующие основные приложения SCM: средство контроля версий PVCS Version Manager; PVCS Tracker — система отслеживания изменений и дефектов; PVCS Configuration Builder, используемый для того, чтобы упростить развертывание новых вариантов. Эти продукты можно приобрести по отдельности или в пакете PVCS Professional Plus.

PVCS Version Manager 6.5, выпущенный в марте, отличается полностью переработанным графическим пользовательским интерфейсом, аналогичным интерфейсу Microsoft Windows Explorer. Version Manager отображает проекты в виде древовидной структуры с левой стороны экрана, а в двух окнах справа выводится содержимое каждой ветви и состояние версии каждого компонента.

Version Manager 6.5, как и предыдущие версии, работает с популярными интегрированными средами разработки и предоставляет прикладные программные интерфейсы для связи с разработанными потребителями инструментальными средствами. Он предоставляет удаленным пользователям доступ к клиентам на базе Web и отличается большей простотой использования и гибкостью, чем его предшественники.

Для того чтобы иметь возможность отслеживать и контролировать запросы на изменения, такие как запрос на новую функцию или исправление обнаруженной ошибки, понадобится PVCS Tracker. Вопросы по поводу изменений возникают во многих проектах, поэтому следует продумать возможность применения Tracker в качестве дополнения к Version Manager.

**ClearCase (Rational Software)**

Компания Rational Software является лидером в области создания методологий и программных решений, ориентированных на программистов, аналитиков, специалистов по тестированию приложений. Спектр выпускаемого обеспечения охватывает потребности всех участников проекта - от аналитиков до разработчиков и сотрудников, занимающихся внедрением готового продукта. Все программно-методологические решения - результат многолетнего труда аналитиков и разработчиков как самой компании Rational, так и ее партнеров и клиентов.

Все эти решения в совокупности составляют RUP (Rational Unified Process) - методологическую энциклопедию, в которой описаны все этапы создания качественного программного продукта. Пользуясь подобной энциклопедией и применяя соответствующие инструменты, рекомендуемые Rational, можно создавать программные продукты качественно и в срок.

Особое место в RUP занимает SCM (Source Code Management) - управление исходным текстом. SCM описывает способ контроля и сопровождения информации, составляющей программный проект. SCM - это методология, которую поддерживает продукт ClearCase, предназначенный для отслеживания и детального протоколирования всего, что связано с хранением всех артефактов, сопровождающих проект (здесь и далее термин "артефакт" следует трактовать как "хранимый документ". Работая над проектом, каждый участник создает определенный набор файлов-артефактов: документов, исходных текстов, бинарных файлов и т.д.).

При коллективной разработке проекта ClearCase рекомендован тем участникам проекта, которые должны обмениваться информацией с другими и точно знать, кто из коллег и когда вносил изменения. Ведь в современных условиях создания приложений большими командами разработчиков просто невозможно обойтись без надежного и мощного средства отслеживания изменений, позволяющего всем участникам команды представлять себе текущее положение дел в разработке проекта.

**Контрольные вопросы:**

1. Конфигурация программного обеспечения (ПО) — это процесс настройки и персонализации параметров программного обеспечения для удовлетворения специфических потребностей пользователя или организации. Это включает в себя выбор определенных настроек, параметров, модулей и компонентов, которые определяют, как ПО будет работать и какие функции оно будет предоставлять.
2. Конфигурационное управление (Configuration Management, CM) — это дисциплина, которая охватывает все аспекты идентификации, контроля, статуса и аудита изменений в конфигурации системы. Это относится не только к программному обеспечению, но и к аппаратному обеспечению, документации и процессам. Цель конфигурационного управления – обеспечить, чтобы система оставалась стабильной, предсказуемой и управляемой на протяжении всего её жизненного цикла.
3. Конфигурационное управление может быть:

* Централизованное: Все данные хранятся на центральном сервере, что упрощает контроль версий, но может быть менее гибким для распределенных команд. Пример: SVN.
* Распределённое: Копии репозиториев хранятся на разных машинах, позволяя автономную работу и гибкость. Пример: Git.
* Потоковое: Управление версиями строится вокруг ветвей и потоков разработки. Подходит для больших проектов с параллельной разработкой. Пример: ClearCase (в некоторых своих аспектах).

1. Процесс управления конфигурацией необходим для:

* Контроля версий: Отслеживание изменений, откат к предыдущим версиям.
* Управления изменениями: Регламентация внесения изменений, одобрение и внедрение изменений.
* Обеспечения целостности: Гарантия того, что все компоненты ПО согласованы между собой.
* Упрощения совместной работы: Позволяет нескольким разработчикам работать над одним проектом одновременно.
* Повышения качества: Позволяет лучше контролировать и отслеживать дефекты.
* Управления рисками: Снижает риски, связанные с внесением изменений и интеграцией компонентов.
* Сокращения времени на тестирование: Использование управляемых версий и автоматизированного тестирования.
* Упрощения процесса выпуска релизов: Контролируемый процесс сборки и выпуска новых версий.

1. Основные этапы процесса управления конфигурацией:

* Идентификация: Определение всех компонентов ПО и их взаимосвязей.
* Контроль: Управление изменениями, одобрение и внедрение.
* Статус: Отслеживание состояния компонентов и изменений.
* Аудит: Проверка соответствия конфигурации установленным требованиям.
* Верификация: Проверка правильности и соответствия всех компонентов.

1. Средства управления конфигурацией (Configuration Management tools) можно разделить на несколько категорий, в зависимости от их функциональности и области применения. Вот некоторые из них:

1. Системы контроля версий (Version Control Systems, VCS): Эти инструменты предназначены для отслеживания изменений в исходном коде и других файлах. Они позволяют хранить историю изменений, отменять изменения, работать в команде над одним проектом и управлять различными версиями.

\* Git: Самая популярная система контроля версий, используемая для разработки программного обеспечения.

\* Subversion (SVN): Более старая, но всё ещё распространённая централизованная система контроля версий.

\* Mercurial (Hg): Распределённая система контроля версий, альтернатива Git.

\* CVS: Одна из первых систем контроля версий, сейчас используется редко.

2. Системы управления изменениями (Change Management Systems): Эти инструменты помогают управлять процессом запроса, утверждения и внедрения изменений в системе. Они обеспечивают отслеживание изменений, уведомления, и контроль за выполнением.

\* Jira: Популярный инструмент для управления проектами и задачами, часто используется для управления изменениями.

\* ServiceNow: Платформа для управления ИТ-сервисами, включающая функционал управления изменениями.

\* Azure DevOps: Платформа Microsoft для управления разработкой и развертыванием программного обеспечения.

3. Инструменты автоматизации инфраструктуры (Infrastructure as Code, IaC): Эти инструменты позволяют описывать и управлять инфраструктурой (серверы, сети, базы данных) с помощью кода. Это обеспечивает повторяемость, автоматизацию и управление версиями инфраструктуры.

\* Terraform: Популярный инструмент IaC от HashiCorp.

\* Ansible: Инструмент IaC, основанный на использовании агентов, упрощающий автоматизацию конфигурации.

\* Chef: Инструмент IaC, использующий рецепты (recipes) для управления конфигурацией.

\* Puppet: Инструмент IaC, основанный на декларативном подходе.

\* CloudFormation (AWS): Сервис AWS для автоматизации развертывания и управления облачной инфраструктурой.

\* Azure Resource Manager (ARM): Сервис Azure для управления ресурсами в облаке Azure.

4. Системы управления конфигурацией (Configuration Management Systems, CMS): Эти инструменты предназначены для автоматизации настройки и управления конфигурацией серверов и других компонентов ИТ-инфраструктуры.

\* Puppet: (уже упомянуто в IaC, но также относится к CMS)

\* Chef: (уже упомянуто в IaC, но также относится к CMS)

\* Ansible: (уже упомянуто в IaC, но также относится к CMS)

\* SaltStack: Система управления конфигурацией, ориентированная на скорость и масштабируемость.

5. Системы управления базами данных конфигурации (Configuration Management Databases, CMDB): Эти инструменты используются для хранения и управления данными о конфигурации системы.

\* ServiceNow: (уже упомянуто, имеет встроенную CMDB)

\* BMC Helix: Платформа для управления ИТ-сервисами, включающая CMDB.

**Вывод:** в ходе выполнения данной лабораторной работы получил навыки произведения настройки и конфигурирования ПО КС.