Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Робототехники и комплексной автоматизации

КАФЕДРА Системы автоматизированного проектирования (РК-6)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка программной технологии автоматизации построения дистрибутивов решателей, построенных на основе графоориетированной технологии

Студент _____ _____ _____
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель _____ _____
(Подпись, дата) (И.О.Фамилия)

2019 г.

СОДЕРЖАНИЕ

СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ

РВС GCD – распределенная вычислительная система GCD;

GUI – графический пользовательский интерфейс ;

CI/CD – непрерывная интеграция;

Пользователь – разработчик решателей для системы GCD;

Решатель – модуль системы GCD, реализующий решение определённого класса задач инженерного анализа;

Git - Распределённая система управления версиями;

SVN - Свободная централизованная система управления версиями;

aINI – Advanced INI – формат представления данных[9];

АННОТАЦИЯ

В рамках данной курсовой работы был разработан плагин для сервера приложений РВС GCD на языке python, позволяющий загружать пользовательские решатели в систему GCD и оптимизировать процесс совместной разработки модулей и решателей.

ВВЕДЕНИЕ

В процессе создания вычислительных и информационных систем более, чем одним разработчиком возникает проблема интеграции различных модулей этого приложения, написанных разными людьми. Также, при параллельной разработке систем, решающих одинаковые задачи или задачи похожего класса различными группами разработчиков было бы намного удобнее иметь возможность публиковать свои реализации на одной платформе, к которой имеют доступ все группы разработчиков. Для решения подобных проблем были созданы системы непрерывной интеграции, позволяющие аккумулировать результаты трудов различных разработчиков программ и систем.

Непрерывная интеграция (*CI*, англ. *Continuous Integration*) — практика разработки программного обеспечения, которая заключается в постоянном слиянии рабочих копий в общую основную ветвь разработки (до нескольких раз в день) и выполнении частых автоматизированных сборок проекта для скорейшего выявления потенциальных дефектов и решения интеграционных проблем. В обычном проекте, где над разными частями системы разработчики трудятся независимо, стадия интеграции является заключительной. Она может непредсказуемо задержать окончание работ. Переход к непрерывной интеграции позволяет снизить трудоёмкость интеграции и сделать её более предсказуемой за счёт наиболее раннего обнаружения и устранения ошибок и противоречий, но основным преимуществом является сокращение стоимости исправления дефекта, за счёт раннего его выявления.[1] Впервые идея была предложена Г.Бучем в 1991 году.[2] С тех пор системы непрерывной интеграции были реализованы различными крупными сервисами и зачастую встроены в Системы контроля версий.

Основные принципы непрерывной интеграции[3]:

Первый принцип: сегрегация ответственности заинтересованных сторон. Одним из основных преимуществ CI/CD является своевременное участие различных заинтересованных сторон в любом проекте.

Второй принцип : снижение риска. Каждый этап конвейерной обработки CI/CD создается для снижения риска в определенном аспекте. Разработчики отвечают за логические и письменные тесты, чтобы снизить риск нарушения логики.

Третий принцип : короткий цикл обратной связи. Причина внедрения конвейерной обработки CI/CD — использование машин для работы с людьми. Это позволяет сократить время, затрачиваемое на обратную связь по разрабатываемым функциям.

На основе этих принципов были описаны варианты организации работы систем с несколькими разработчиками на основе системы контроля версий git[4,5,6] или на основе собственных реализаций[7,8].

В процессе развития механизма непрерывной интеграции были реализованы различные подходы. Наиболее современные и перспективные из них[3]:

1) Локальные:
   a) GitLab CI – интегрированная в сервис GitLab система CI. Выполняет валидацию файлов перед их добавлением в проект.
   b) TeamCity, Bamboo, Cirkle CI, Jenkins - это многофункциональный серверы непрерывной интеграции, готовые к работе сразу же после установки. Он поддерживают множество систем контроля версий, аутентификации, сборки и тестирования прямо из коробки.
2) Облачные:

BitBucket Pipelines, Heroku CI, Travis, Codeship, Buddy CI, AWS CodeBuild – Системы для проектов, которые не требуют локального хостинга и, зачастую, с открытым исходным кодом, имеют между собой небольшие различия в реализуемых функциях.

Таким образом принципы технологии непрерывной интеграции стоит использовать при разработке системы автоматизированного построения дистрибутивов решателей.

1. ПОСТАНОВКА ЗАДАЧИ

В рамках данной курсовой работы необходимо разработать плагин для сервера приложений GCD на языке python для копирования скомпилированных файлов решателей и файлов параметров в формате aINI из git репозитория разработчиков в git репозитории PBC GCD, на основе которых решатели включаются в сборку системы отображаются в её GUI.

2. АРХИТЕКТУРА

1) Формат приложения – плагин на языке python

В рамках реализуемой системы удобно реализовать плагин, интегрированный в сервер приложений. В таком случае приложение получает доступ к БД системы и списку зарегистрированных плагинов, а также может получает возможность самостоятельно регистрировать новые плагины после их валидации.

2) Язык программирования – python

Программа написана на языке python. Python — это удобный высокоуровневый язык программирования, имеющий в своих стандартных библиотеках

огромное количество инструментов, позволяющих реализовать любой функционал приложения.

Выбор данного языка обусловлен тем, что в нем есть все необходимые, для реализации, инструменты. В нем очень удобно работать со строками, что необходимо при генерации путей расположения и имён файлов. Также в языке просто взаимодействовать с операционной системой и git репозиториями.

3) Используемая CI/CD – GitLab CI

GitLab CI позволяет встроенными средствами автоматизировать процесс проверки и интеграции в репозитории уже существующего работающего проекта новых модулей из репозиториев разработчиков.

3. ПРИНЦИП РАБОТЫ ПРИЛОЖЕНИЯ

1) Общий принцип работы приложения

Блок-схема работы программы представлена на Рис.1. Разработанное последовательное функционирование программы позволяет в будущем применить графоориентированную технологию [10] к работе программы. Применение графоориентированного подхода позволит гибко заменять и/или обновлять любые функции-обработчики, а также естественным образом формировать библиотеки функций программного инструментария в целом.

Схема работы пользователей с программой и программы с данными представлена на Рис. 2. Все данные, используемые программой должен настраивать разработчик решателя перед запуском программы.

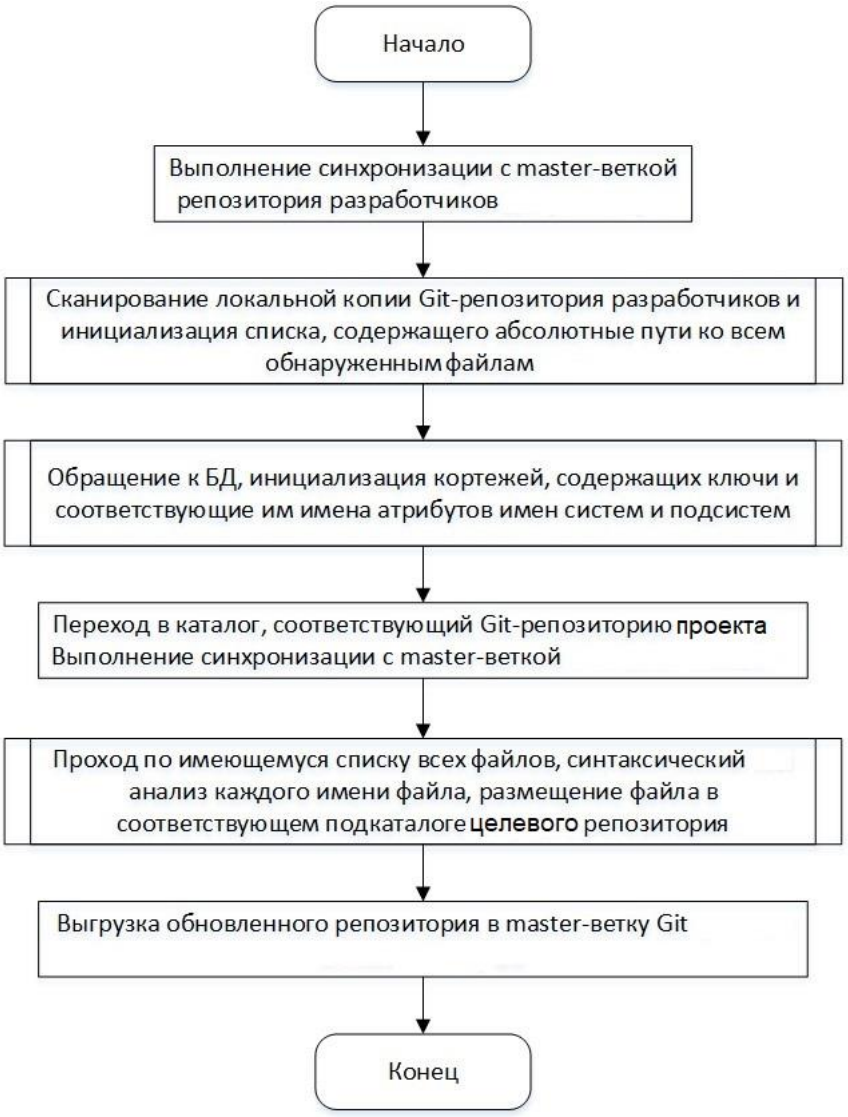

Рис 1. Блок-схема разработанной программы

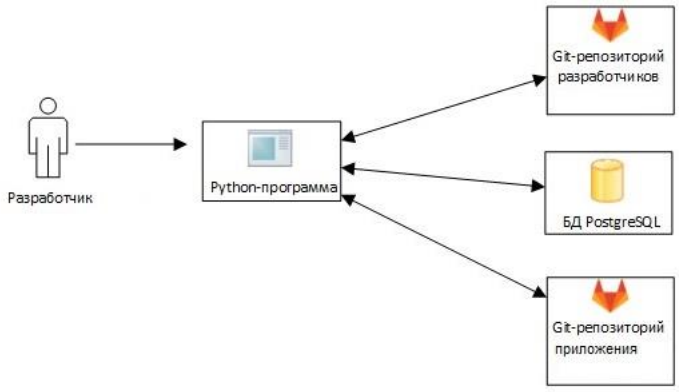

Рис 2. Схема взаимодействия программы с внешними данными и пользователями

2) Сценарий работы плагина

Разработчик завершает работу над решателем и запускает плагин с помощью GitLab CI, который просканирует локальный репозиторий разработчика на наличие новых файлов, относящихся к решателям, сравнивая его с master веткой на сервере. Затем, на основе данных о подсистеме GCD, к которой относится решатель, выполняется подключение к репозиториям с решателями текущей версии PBC GCD и синхронизация их с текущей версией master ветки репозитрия разработчиков, генерируется его путь в них, если целевых каталогов не существует – они создаются. После чего все файлы размещаются в соответствии с целевыми для них каталогами и репозиториями, выполняется тестовая сборка обновлённой версии системы GCD и, в случае успешного прохождения тестов выполняется размещение всех изменений в master ветке git репозитория разработки.

ЗАКЛЮЧЕНИЕ

Возможность разработчикам создавать и публиковать в системе GCD свои решатели, используя плагин автоматизированного построения дистрибутивов позволяет существенно сократить время, затраченное на публикацию решателя и его отладку. Это позволит им сосредоточиться на разработке, а также без затруднений делиться своими решателями с другими разработчиками, что делает процесс более удобным и быстрым.

СПИСОК ЛИТЕРАТУРЫ:

1. Continuous integration. Wikipedia [электронный ресурс]. - Режим доступа: https://en.wikipedia.org/wiki/Continuous_integration
2. Booch G. Object Oriented Design: With Applications. — Benjamin Cummings, 1991. — 209 p.
3. That CI/CD Thingy: Principles, Implementation & Tools. Medium [электронный ресурс]– Режим доступа: https://blog.gds-gov.tech/that-ci-cd-thing-principles-implementation-tools-aa8e77f9a350
4. Атаян Б. Г. Метод выявления измененных файлов в облачной системе резервного копирования //Вестник Национального политехнического университета Армении - 2018 - № 1. - С. 20-29.
5. Гаспарян А. В., Тимошина Н. В. Совместная разработка ПО с использованием *Git* //ИТпортал. – 2017. – №. 1 (13).
6. Ерошенко Я. Б., Самхарадзе К. К. Оптимизация сборки программных средств в системе *Git* //Аллея науки. – 2017. – Т. 2. – №. 11. – С. 453-459.
7. Кузьмина И. В., Фидельман В. Р. Разработка программного обеспечения сложных аппаратно-программных комплексов с использованием принципов непрерывной интеграции // Известия ВУЗов. Поволжский регион. Технические науки. - 2012. - №2. – С. 13.
8. Дадыкин А. К., Ермолаев А. А. Современная методика организации процесса разработки программ // Системный анализ и прикладная информатика. - 2013. - №1-2. – С. 23.
9. Соколов А.П., Першин А.Ю. Формат данных Advanced INI (aINI) // Каркас системы – 2007-2017 – SA2 – 18 стр.
10. Соколов А.П., Першин А.Ю., Щетинин В.Н., Сапелкин А.С. Реверсивная многомасштабная гомогенизация физико-механических характеристик гетерогенных периодических сред с использованием графоориентированного программного подхода – Композиты и наноструктуры. 2017, Т.9, № 3-4, с. 25-38.