**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**

[Аннотация 3](#_TOC_250006)

[ВВЕДЕНИЕ 4](#_TOC_250005)

1. [ПОСТАНОВКА ЗАДАЧИ 6](#_TOC_250004)
2. [АРХИТЕКТУРА 6](#_TOC_250003)
3. [ПРИНЦИП РАБОТЫ ПРИЛОЖЕНИЯ 7](#_TOC_250002)

[ЗАКЛЮЧЕНИЕ 10](#_TOC_250001)

[Список литературы 11](#_TOC_250000)

# СОКРАЩЕНИЯ И ОБОЗНАЧЕНИЯ

РВС GCD – Распределенная вычислительная система GCD.

GUI – графический пользовательский интерфейс (англ. Graphical user interface).

CI – непрерывная интеграция (англ. Continuous Integration).

Git - система контроля версий с распределённым типом хранения репозиториев.

SVN - Свободная централизованная система управления версиями.

aINI – формат представления исходных данных (англ. Advanced INI).

# АННОТАЦИЯ

Работа посвящена созданию программной технологии автоматизации построения дистрибутивов «решателей», построенных на основе графоориентированной технологии.

В рамках данной курсовой работы был разработан модуль расширения для Распределённой вычислительной системы GCD на языке Python. Созданный программный модуль позволил осуществлять сканирование некоторого репозитория проекта разработки (в работе рассматривались Git репозитории) нового «решателя» и формирование на его основе пакета файлов, составляющих его дистрибутив, с последующим его архивированием и загрузкой в целевое назначение, заранее указываемое пользователем.

Разработанный модуль расширения позволил автоматизировать процесс создания дистрибутивов решателей и их накопление в доступном для скачивания расположении.

# ВВЕДЕНИЕ

Пользователь – разработчик решателей для системы GCD.

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

После разработки решателя, включая создание графовой модели, определение исходных данных, разработку функций-обработчиков и функций-предикатов должен быть собран дистрибутив нового решателя. Фактически дистрибутив формируется на основе скомпилированных бинарных модулей и сопутствующих файлов ресурсов. Реализация задачи позволит формализовать и автоматизировать процесс построения дистрибутива решателя на основе его графовой модели с формированием инсталлятора, включающего автоматически формируемую документацию.

В процессе создания вычислительных и информационных систем более, чем одним разработчиком возникает проблема интеграции различных моду- лей этого приложения, написанных разными людьми. Также, при параллель- ной 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 аккумулиро- вать результаты трудов различных разработчиков программ и систем.

**Непрерывная интеграция** (*CI*, англ. *Continuous Integration*) — прак- тика разработки программного обеспечения, которая заключается в постоян- ном слиянии рабочих копий в общую основную ветвь разработки (до несколь- ких 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 интегра- ции позволяет снизить трудоёмкость интеграции и сделать её более предска- зуемой за счёт наиболее раннего обнаружения и устранения ошибок и проти- воречий, но основным преимуществом является сокращение стоимости ис- правления дефекта, за счёт раннего его выявления.[1] Впервые идея была предложена Г.Бучем в 1991 году.[2] С тех пор системы непрерывной интегра- ции 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 раз- личных 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 разраба- тываемым функциям.

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

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

1. Локальные:
   1. GitLab CI – интегрированная в сервис GitLab система CI. Выполняет валидацию файлов перед их добавлением в проект.
   2. TeamCity, Bamboo, Cirkle CI, Jenkins - это многофункциональный сер- веры непрерывной интеграции, готовые к работе сразу же после уста- новки. Он поддерживают множество систем контроля версий, аутенти- фикации, 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 хо- стинга и, зачастую, с открытым исходным кодом, имеют между собой небольшие различия в реализуемых функциях.

Таким образом принципы технологии непрерывной интеграции стоит ис- пользовать при разработке системы автоматизированного построения дистри- бутивов решателей.

# ПОСТАНОВКА ЗАДАЧИ

В рамках данной курсовой работы необходимо разработать плагин для сер- вера приложений GCD на языке python для копирования скомпилированных файлов решателей и файлов параметров в формате aINI из git репозитория раз- работчиков в git репозитории PBC GCD, на основе которых решатели включа- ются в сборку системы отображаются в её GUI.

# АРХИТЕКТУРА

1. Формат приложения – плагин на языке python

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

1. Язык программирования – python

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

огромное количество инструментов, позволяющих реализовать любой функ- ционал приложения.

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

1. Используемая CI/CD – GitLab CI

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

# ПРИНЦИП РАБОТЫ ПРИЛОЖЕНИЯ

1. Общий принцип работы приложения

Блок-схема работы программы представлена на Рис.1. Разработанное по- следовательное функционирование программы позволяет в будущем приме- нить графоориентированную технологию [10] к работе программы. Примене- ние графоориентированного подхода позволит гибко заменять и/или обнов- лять любые функции-обработчики, а также естественным образом формиро- вать библиотеки функций программного инструментария в целом.

Схема работы пользователей с программой и программы с данными пред- ставлена на Рис. 2. Все данные, используемые программой должен настраи- вать разработчик решателя перед запуском программы.



Рис 1. Блок-схема разработанной программы

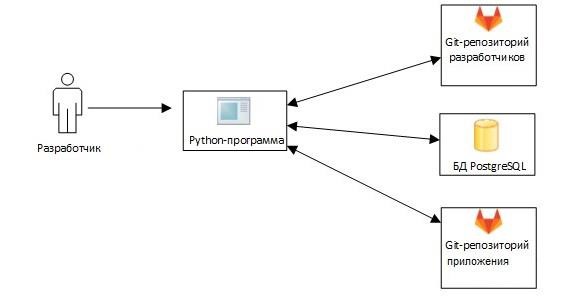


Рис 2. Схема взаимодействия программы с внешними данными и пользователями

1. Сценарий работы плагина

Разработчик завершает работу над решателем и запускает плагин с помо- щью 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 от- носится 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 катало- гов не существует – они создаются. После чего все файлы размещаются в со- ответствии с целевыми для них каталогами и репозиториями, выполняется те- стовая сборка обновлённой версии системы GCD и, в случае успешного про- хождения 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 дистрибу- тивов 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 разработчи- ками, 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 [электрон- ный ресурс]– Режим доступа: https://blog.gds-gov.tech/that-ci-cd-thing-princi- ples-implementation-tools-aa8e77f9a350
4. Атаян Б. Г. Метод выявления измененных файлов в облачной системе ре- зервного копирования //Вестник Национального политехнического универ- ситета Армении - 2018 - № 1. - С. 20-29.
5. Гаспарян А. В., Тимошина Н. В. Совместная разработка ПО с использова- нием *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 принци- пов непрерывной интеграции // Известия ВУЗов. Поволжский регион. Тех- нические науки. - 2012. - №2. – С. 13.
8. Дадыкин А. К., Ермолаев А. А. Современная методика организации про- цесса разработки программ // Системный анализ и прикладная информа- тика. - 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 ге- терогенных периодических сред с использованием графоориентированного программного подхода – Композиты и наноструктуры. 2017, Т.9, № 3-4, с. 25-38.