УДК 004

Применение модуля pyreverse для создания диаграмм классов в проекте Python

Вихляев Дмитрий Романович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье рассматривается метод создания диаграмм классов из исходного кода Python. Для реализации используется модуль pyreverse и программа graphviz. Результатом исследования станет описание визуализации структур как пользовательских, так и библиотеках классов в программе Python.
Ключевые слова: Python, UML, диаграмма классов.

Using the pyreverse module to create class diagrams in a Python project

Vikhlyaev Dmitry Romanovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article discusses a method for creating class diagrams from Python source code. The pyreverse module and the graphviz program are used for implementation. The result of the study will be a description of the visualization of structures in both user-defined and class libraries in the Python program.
Keywords: Python, UML, class diagram.

1 Введение

1.1 Актуальность

Структурный анализ уже написанного программного кода, является неотъемлемой частью для написания документаций. Помимо текстового описания немало важным является графическое описание структур, связанностей и функциональных компонентов. Создание диаграмм всегда находится на переднем плане среди других описательных методов. В начтоящее время существуют уже готовые решения для автоматизации данного процесса. Pyreverse является мощным инструментом для генерации UML диаграмм классов и пакетов из исходного кода Python. Это полезный инструмент для разработки и документирования программного обеспечения, так как он позволяет визуализировать структуру и зависимости между классами. Основные функции включают анализ исходного кода и построение графа зависимостей.

1.2Обзор исследований

С.В.Миронов описала подход к структурному анализу исходных кодов программного обеспечения [1]. А.В.Киселев представил метод тестирования классов объектно-ориентированного программного обеспечения [2]. Г.Н.Катаев в своей статье сделал обзор средств автоматического формирования документации по программному коду [3]. Е.С.Захарова, М.А.Кузнецов исследовали процесс документирования исходного кода при ведении проекта как средство сокращения ресурсов предприятия [4]. Е.В.Долгова, Е.А.Нерослов разработали систему автоматизированной генерации диаграмм классов, схем баз данных, прототипов визуальных макетов на основе анализа текста на естественном языке [5].

1.3Цель исследования

Цель исследования – описать способ создания диаграмм классов с помощью инструмента Pyreverse.

2 Материалы и методы

Для работы используется программный модуль Python Pylint, примеры на исходных кодах Python и дополнительная программа graphviz для генерации изображений в формате PNG.

3 Результаты и обсуждения

Pyreverse входит в состав пакета Pylint, который можно установить с помощью менеджера пакетов pip. Имеется поддержка различных форматов вывода, включая dot, puml, plantuml, mmd, html.

По умолчанию используется формат dot. Это специфичный язык, с помощью которого можно проектировать различные диаграммы, включая UML. Язык поддерживает программа редактор graphviz. Если загрузить данное приложение и установить путь в переменную среды к папке с исполняемыми файлами, то в Pyreverse можно указать вывод в формате PNG. Описанный метод используется для приведения примеров в данном исследовании.

Pyreverse поддерживает несколько команд и опций, которые позволяют настроить процесс анализа и генерации диаграмм. В представленных примерах будет использоваться аргумент «-o <format>», который указывает формат выходного файла диаграммы. Аргументы уровней «-a <level>» и «-s <level>», устанавливают уровень анализа зависимостей (например, -a1 для анализа явных зависимостей) и уровень сложности диаграммы (например, -s1 для отображения только наследования и реализации). Чтобы анализировать только указанные модули исполбзуется «-mn <module>», или фильтры для включения или исключения элементов «-f <filter>» (например, -f ALL для анализа всех элементов) На рисунке 1 представлен классический пример построения связанных классов. Используя команду «Pyreverse -o png path_file», в текущей директории создаться изображение с диаграммой классов с указанием связей (рис.2).

```python
class Animal:
    def speak(self):
        pass

class Dog(Animal):
    def speak(self):
        return "Bark"

# Реализация (Implementation)
class Eatable:
    def eat(self):
        pass

class Apple(Eatable):
    def eat(self):
        return "Eat apple"

# Ассоциация (Association)
class Pet:
    def __init__(self, name):
        self.name = name

class Person:
    def __init__(self, name):
        self.name = name
        self.pet = None  # Ассоциация с классом Pet

    def adopt_pet(self, pet: Pet):
        self.pet = pet

# Композиция (Composition)
class Engine:
    def start(self):
        return "Engine starts"

class Car:
    def __init__(self):
        self.engine = Engine()  # Композиция: Car содержит Engine

    def start(self):
        return self.engine.start()

# Зависимость (Dependency)
class Database:
    def connect(self):
        return "Database connected"

class Application:
    def __init__(self, db: Database):
        self.db = db

    def start(self):
        return self.db.connect()
```

Рис. 1. Реализация классов

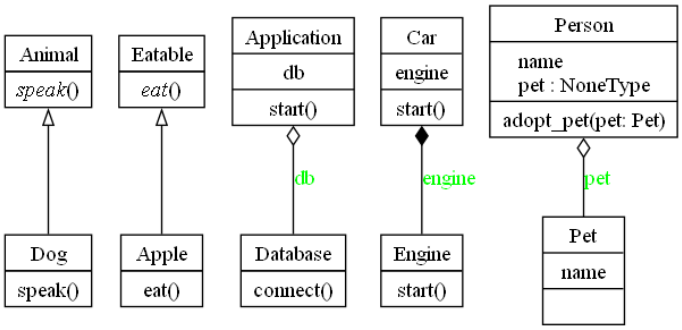

Рис. 2. Сгенерированная диаграмма

Если имеется директория с несколькими файлами, то можно загрузить все модули за раз как один проект. Для этого можно использовать команду «pyreverse -o png -p MyProject path_direct». Тогда помимо классов появится диаграмма связи модулей (рис.3,4,5).

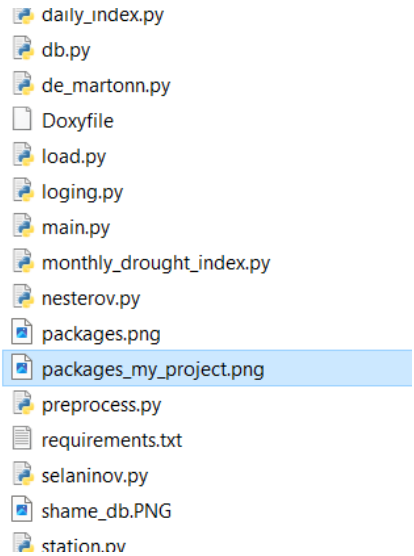

Рис. 3. Содержание директории проекта

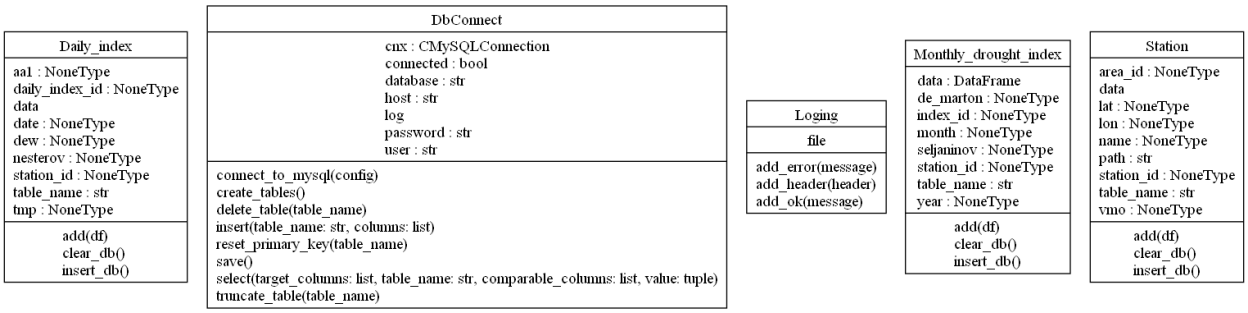

Рис. 4. Диаграмма классов проекта

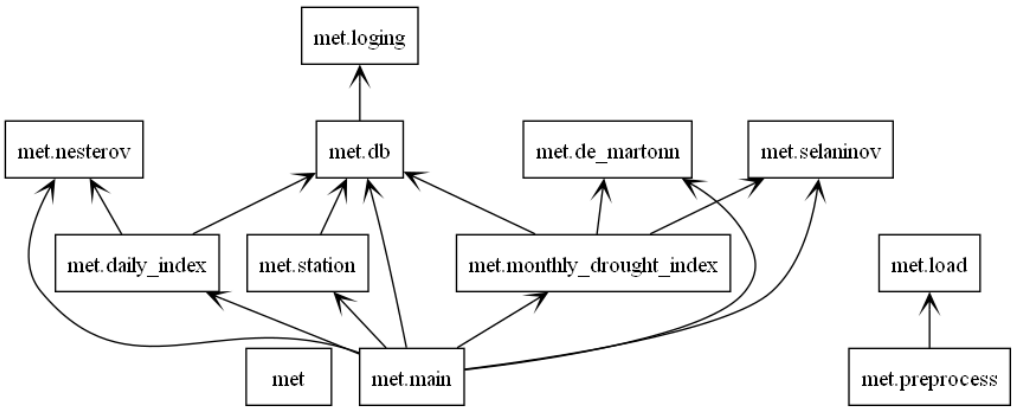

Рис. 5. Диаграмма связей модулей

Инструмент Pyreverse позволяет исследовать не только пользовательские классы и модули. В проекте на Python, всегда имеется большое количество подключаемых библиотек. Чтобы посмотреть взаимосвязь пользовательского класса со структурами из подключенного модуля необходимо изменять уровень сложности диаграммы посредством аргумента (s0-s3). Для описания используется модуль, взаимодействующий с базой данных через «mysql.connector» На рисунке 6 представлена часть кода с объявлением и функцией инициализации класса, а также метод подключения к базе данных. При использовании нулевого уровня (по

умолчанию), на диаграмме отобразится только пользовательский класс, результат на рисунке 7. В первом уровне отображаются классы, напрямую используемые в пользовательской структуре. Соответственно второй уровень спускается на шаг, ниже предоставляя пользователю внутреннее устройство подключаемого модуля (рис.8,9).

```python
import mysql.connector
from mysql.connector import errorcode
from datetime import datetime
from loging import Loging

class DbConnect():
    def __init__(self):
        self.user="mysql"
        self.password="mysql"
        self.database="test"
        self.host="localhost"

        config = {
            "host": self.host,
            "user": self.user,
            "password": self.password,
            "database": self.database,
        }

        self.log=Loging()
        self.log.add_header("CONNECT TO DB")
        self.connected=False
        self.cnx=self.connect_to_mysql(config)
        if self.cnx and self.cnx.is_connected():
            self.log.add_ok("Connected success")
            self.connected=True



    def connect_to_mysql(self,config):
        try:
            return mysql.connector.connect(**config)
        except mysql.connector.Error as err:
            if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
                self.log.add_error("wrong login or password")
                print("Something is wrong with your user name or password")
            elif err.errno == errorcode.ER_BAD_DB_ERROR:
                self.log.add_error("Database does not exist")
                print("Database does not exist")
            else:
                print(err)
                self.log.add_error(err)
            return None
```

Рис. 6. Реализация класса подключения к базе данных

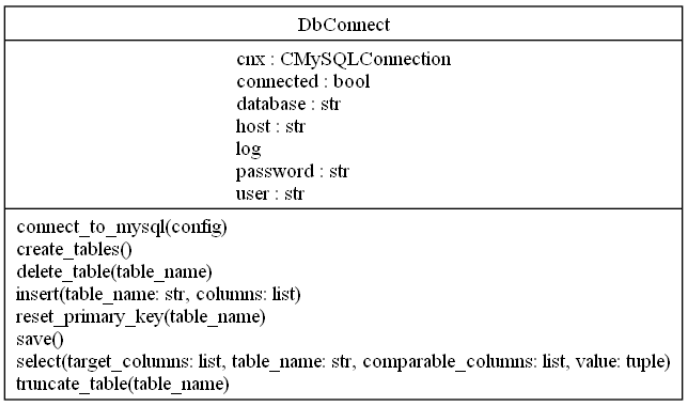

Рис. 7. Диаграмма классов нулевого уровня сложности

Рис. 8. Диаграмма классов первого уровня сложности

Рис. 9. Диаграмма классов второго уровня сложности

Таким образом, можно утверждать, что Pyreverse является полезным инструментом для Python разработчиков. С его помощью можно как документировать созданные проекты, так и проводить анализ стороннего кода.

В результате исследования был рассмотрен инструмент Pyreverse, предназначенный для структурного анализа кода на языке программирования Python. Приведены примеры использования и описаны основные команды для начинающих пользователей.

Библиографический список

1. Миронов С.В. Подход к структурному анализу исходных кодов программного обеспечения // В сборнике: Безопасные информационные технологии. Сборник трудов Восьмой всероссийской научно-технической конференции. НУК «Информатика и системы управления». Под. ред. М.А.Басараба. 2017. С. 308-310.
2. Киселев А.В. Метод тестирования классов объектно-ориентированного программного обеспечения // Вестник Нижегородского университета им. Н.И. Лобачевского. 2012. № 2-1. С. 210-215.
3. Катаев Г.Н. Обзор средств автоматического формирования документации по программному коду // Научный аспект. 2024. Т. 5. № 4. С. 568-578.
4. Захарова Е.С., Кузнецов М.А. Внедрение процесса документирования исходного кода при ведении проекта как средство сокращения ресурсов предприятия // В сборнике: Цифровая трансформация общества, экономики, менеджмента и образования. Материалы III Международной конференции. 2020. С. 75-80.
5. Долгова Е.В., Неростов Е.А. Разработка системы автоматизированной генерации диаграмм классов, схем баз данных, прототипов визуальных макетов на основе анализа текста на естественном языке // В сборнике: Автоматизированные системы управления и информационные технологии. Материалы всероссийской научно-технической конференции. Пермь, 2023. С. 240-245.
6. Поликарпова А.И., Самочадин А.В. Система реинжиниринга для

генерации uml-документации кода x++ // В сборнике: Современные технологии в теории и практике программирования. Сборник материалов научно-практической конференции студентов, аспирантов и молодых ученых. Санкт-Петербург, 2024. С. 251-252.

7. Синицына Т.И. Методы проверки программной документации // В сборнике: Математика и междисциплинарные исследования - 2018. Материалы Всероссийской научно-практической конференции молодых ученых с международным участием. 2018. С. 92-95.
8. Денисов А.М., Макаров В.В. Опыт автоматизации процессов проектирования, тестирования и документирования программного обеспечения технических систем реального времени // В сборнике: Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта (cad/cam/pdm - 2010). Труды международной конференции. 2010. с. 89-93.