Retaining maintainability throughout the scaling of software projects using automated tools

Deutscher Titel

Nikkel Mollenhauer

Universitätsbachelorarbeit
zur Erlangung des akademischen Grades

Bachelor of Science
(B. Sc.)

im Studiengang
IT-Systems Engineering

eingereicht am 30. Juni 2022 am
Fachgebiet Enterprise Platform and Integration Concepts der
Digital-Engineering-Fakultät
der Universität Potsdam

Gutachter	Dr. Michael Perscheid
Betreuer	Rainer Schlosser
	Johannes Huegle
	Alexander Kastius

Abstract

When scaling software projects, knowing when and how to start scaling maintainability is the key to success. Maintainability manifests in many different ways, be it well documented API's, a comprehensive wiki or sensible tests. Maintainability can be aided by automating many processes, as this takes strain away from developers and lowers the barrier of entry for new team members looking to contribute.

In our project, we employed many of these techniques, and this thesis aims to illustrate the goals and ideas behind the processes involved. Primary focus is the journey from a single-file project to the pip-package 'recommerce' with its automated testing pipeline, code-style checks and comprehensive documentation using automated tools such as 'Sphinx' and 'interrogate'.

Analysis of the process shows that while tests are useful in aiding developers to understand code it must always be kept in mind that while tests make certain aspects of the software development process more accessible, they are also to be maintained.

One of the biggest and most costly undertakings was the introduction of pip-packaging the project, which leads to the conclusion that while building and maintaining a clear vision of the desired state of the product as early as possible can lead to an overhead early in the process, it will also make the project grow more sustainably, leading to higher maintainability in the long run.

Problem

Background

Objective

Methodology

Results - focuses on tests while leaving out documentation and pip (which comes in conclusion?!?)

conclusion

Zusammenfassung

Acknowledgments

Contents

Part I

Introduction

1 Knowing your Users(?)

1.1 Feather by SAP

1.2 Alex == Research

2 Related Work(?)

2.1 Packaging discussion in the Python community

Part II

Challenges

3 Documentation

3.1 Wiki

3.2 Docstrings

3.3 Tests

4 Tests

4.1 Pytest

4.1.1 Setup & Teardown == Automation

4.1.2 Parametrization

4.2 What to test

4.3 Types of tests (e.g. Unit vs. Acceptance)

4.4 Writing maintainable tests

4.5 Measuring Coverage

Part III

Automation Tools

5 Pre-commit

5.1 Flake8

5.2 Interrogate

6 Github Actions

7 Sphinx/Readthedocs

Part IV

Packaging the Project

8 Why package the project?

9 Using Pip

10 Conclusions & Outlook

Bibliography

Declaration of Authorship

I hereby declare that this thesis is my own unaided work. All direct or indirect sources used are acknowledged as references.

Potsdam, 22nd March 2022

Nikkel Mollenhauer