Hochschule für angewandte Wissenschaften Augsburg
Fakultät für Informatik

Bachelorarbeit

Vergleich von Python Test-Tools für Test-driven development

zur Erlangung des akademischen Grades
Bachelor of Science

Thema:	Vergleich von Python Test-Tools für Test-driven development
Autor:	Maximilian Konter maximilian.konter@hs-augsburg.de MatNr. 951004
Version vom:	14. März 2019
1. Betreuer:	Dipl.-Inf. (FH), Dipl.-De Erich Seifert, MA
2. BetreuerIn:	Prof. Dr. X

Abstract kommt hier hin

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listingverzeichnis

Glossar

commit Ein commit ist die Sammlung von änderungen an Dateien, welche mithilfe eines VCS verwaltet werden. 12, 13

mock Etwas mocken bedeutet, ein Objekt durch ein falsches Objekt, den Mock zu ersetzen, das genau so aussieht, wie das erwartete Objekt, aber nicht das Gleiche ist. 11, 13

Versions Control System Ein VCS ist ein Dienst, der es seinen Nutzern ermöglicht änderungen an Dateien in einer Chronik zu speichern um später darauf zu zu greifen. Dies dient auch der verteilung von Daten über mehrere Systeme sowie sicherung der Daten vor verlust.. 10

Abkürzungsverzeichnis

GNU	GNU is not Unix
GPL	GNU General Public License
GUI	Graphical User Interface
LGPL	GNU Lesser General Public License
STDLIB	Python Standard Bibliothek
TDD	Test-driven development
VCS	Versions Control System

1 Einleitung

Das ist meine Einleitung

1.1 Was ist Python

Blubb

1.2 Was ist Test-driven development

Blubb

2 Python Test-Tools

Dieses Kapitel befasst sich mit den von der STDLIB bereitgestellten Test-Tools sowie denen aus externen Paketen. Diese werden unter 2.1 und 2.2 zusammengefasst, wobei die Tools aus externen Paketen aufgeteilt werden in Python Test-Tools, Python-Framework Test-Tools und Sonstige Tools.

Jedes Tool wird anhand folgender Aspekte untersucht:

- Anwendbarkeit:
  Bietet das Tool alles, um TDD betreiben zu können? (Bei TDD kann man Tests mocken, wodurch sie nicht fehlschlagen.) Mit wie vielen Paketen muss das Tool betrieben werden?

- Effizienz:
  Wie viel lässt sich mit diesem Tool möglichst einfach und schnell erreichen? Ist besonders viel Vorarbeit notwendig um die Tests auf zu setzen oder kann sofort mit dem Schreiben der Tests begonnen werden?
  Genauso stellt sich die Frage, wie Effizient der Entwickler die Tests auswerten kann.

- Komplexität:
  Wie komplex ist das Tool? Das heißt, wie viel Funktionalität bietet das Tool dem Entwickler von Haus aus, aber auch wie schwer ist es einen Code zu schreiben oder wie schnell wird ein Code unübersichtlich, da das Tool viel Code abseits der Tests benötigt.

- Erweiterbarkeit:
  Wie leicht lässt sich das Tool mit anderen Tools erweitern? Gibt es vielleicht Erweiterungen der Community für dieses Tool, die sehr hilfreich sind?

2.1 Welche Tools bietet die Standard Bibliothek von Python?

Die Standard Bibliothek von Python bietet zwei verschiedene Test-Tools(Muthukadan et al., 2011). Zum einen ist dies unittest[1] und zum anderen doctest[2]. Diese beiden Tools reichen sind im ihrem Umfang bereits so vielseitig dass, es einfach ist eine hohe Test-Abdeckung eines Programms oder eine Bibliothek zu erreichen.

Beide Tools zählen zu den „Unit Testing Tools“(Muthukadan et al., 2011) - auf Deutsch Modul Test-Tools - mit deren Hilfe die einzelnen Module eines Programms getestet werden können. In einem Programm oder einer Bibliothek wären dies die einzelnen Funktionen und Methoden.

2.2 Welche Tools gibt es abseits der Standard Bibliothek?

Alle tools abseits der stdlib

2.2.1 Python Test-Tools

Auf https://wiki.python.org/moin/PythonTestingToolsTaxonomy werden viele externe Tools gelistet, jedoch scheinen viele inaktiv zu sein da ihre commits teilweise mehr als ein Jahr zurück liegen. Dies ist weder für eine Bibliothek noch für ein Tool ein gutes Zeichen, da sich die Anforderungen stetig ändern und niemals alle Bugs gefixt sind.

Manche der dort aufgelisteten Tools sind bereits oder werden gerade in andere Tools integriert. Dies kann zum einem sein, da ein Tool eine Erweiterung für ein anderes war und die Entwickler die Änderungen angenommen haben und zum anderen um die Tools zu verbessern und mehr Entwickler zur Verfügung zu haben. Eventuell sind auch andere Gründe dafür verantwortlich, jedoch war dies der Grund bei zum Beispiel Testify von Yelp [3].

[1] http://pyunit.sourceforge.net/pyunit.html
[2] https://docs.python.org/3/library/doctest.html
[3] https://github.com/Yelp/Testify/

Lässt man die Erweiterungen von Tool zunächst außen vor und ignoriert Tools deren letzter commit älter als ein Jahr ist, so bleiben nach Muthukadan et al., 2011 folgende Modul Test-Tools zur Verfügung:

- py.test[4]
- nose[5]
- tofu[6]
- zope.testing[7]

Als weitere Kategorie werden Mock-Tools geführt. Zumeist erweitern diese bereits bestehende Tools wie zum Beispiel

2.2.2 Python-Framework Test-Tools

Blubb

2.2.3 Sonstige Tools

Blubb

2.3 unittest

Blubb

3 Zusammenfassung

Alle Tools hier

3.1 unittest

Blubb

[4] http://pytest.org/latest/
[5] http://pytest.org/latest/
[6] https://www.reahl.org/docs/4.0/devtools/tofu.d.html
[7] https://pypi.org/project/zope.testing/

4 Vergleich der Tools

Blubb

5 Kombinierung von Tools

Blubb

6 Diskussion: Lohnt sich Test-driven development

Blubb

6.1 Welche Stärken hat Test-driven development?

Blubb

6.2 Welche Schwächen hat Test-driven development?

Blubb

6.2.1 Welche Möglichkeiten gibt es diese Schwächen zu umgehen?

Blubb

6.2.2 Welche Schwächen mussen akzeptiert werden?

Blubb

6.3 Welche wirtschaftlichen Aspekte müssen bei Test-driven development beachtet werden?

Blubb

6.4 Zusammenfassung

Blubb

7 Fazit

FAZIT

8 Nachwort

Nachwort

Literaturverzeichnis

Muthukadan, B., jtatum, CPE0014bf07ffd2-CM001ac30d4aca, 71-35-143-156, gjb1002, little-black-box, ... SteveBarnes. (2011). PythonTestingToolsTaxonomy. Website. Online erhältlich unter https://wiki.python.org/moin/PythonTestingToolsTaxonomy; abgerufen am 14. März 2019.

Eidesstattliche Erklärung

Eidesstattliche Erklärung zur Abschlussarbeit

Hiermit versichere ich, die eingereichte Abschlussarbeit selbständig verfasst und keine andere als die von mir angegebenen Quellen und Hilfsmittel benutzt zu haben. Wörtlich oder inhaltlich verwendete Quellen wurden entsprechend den anerkannten Regeln wissenschaftlichen Arbeitens zitiert. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht anderweitig als Abschlussarbeit eingereicht wurde.

Das Merkblatt zum Täuschungsverbot im Prüfungsverfahren der Hochschule Augsburg habe ich gelesen und zur Kenntnis genommen. Ich versichere, dass die von mir abgegebene Arbeit keinerlei Plagiate, Texte oder Bilder umfasst, die durch von mir beauftragte Dritte erstellt wurden.

$Unterschrift:$ $Ort, Datum:$