第 1 卷◆第 1 期◆版本 1.0◆2023 年
文章类型：论文|刊号（ISSN）：2972-4236(P) / 2972-4244(O)

# 基于 Python 敏捷开发设计策略分析

王绎茗
北京二十一世纪国际学校
DOI:10.12238/acair.v1i1.6129

[摘　要] 在基于Python的开发中,该语言具有易用、灵活性和强大的库。并且基于Python的敏捷开发可以成为一种有效的软件开发方法,为团队提供了快速迭代代码和交付高质量软件产品所需的灵活性和协作。
[关键词] Python；敏捷开发；策略
中图分类号：TD219　文献标识码：A

## Analysis of Agile Development Design Strategy Based on Python

Yiming Wang
Beijing 21st Century International School

[Abstract] In Python based development, this approach may be particularly effective due to the language's ease of use, flexibility, and powerful libraries. Python based agile development can become an effective software development method, providing teams with the flexibility and collaboration needed to quickly iterate code and deliver high-quality software products.
[Key words] Python; agile development; strategy

## 引言

敏捷开发是新一代软件开发的重要方法,其具有易用、灵活性和强大的库。基于Python进行敏捷开发能够节省时间和金钱与不必要的工作。因此为了发挥其作用,本文就基于Python敏捷开发设计策略进行了探讨分析。

## 1 敏捷开发的基本概念

在基于Python的开发中,可以通过使用Git等工具进行版本控制,Slack进行实时沟通,以及GitHub或Bitbucket等代码审查工具进行同行审查来实现。

## 2 传统开发模式与敏捷开发的对比

2.1瀑布模型

瀑布模型是一种起源于20世纪70年代的软件开发方法。它是一种连续的软件开发方法,由几个不同的阶段组成,每个阶段都必须在下一个阶段开始之前完成。

瀑布模型的各个阶段通常如下：

需求收集和分析,在这个阶段,对软件的需求进行收集和分析,以确定软件需要做什么。

设计,接下来,在需求的基础上创建软件设计。这个阶段涉及到为如何构建软件创建一个详细的计划,包括架构、软件模块和接口。

实施,在实施阶段,软件将被实际构建。这通常包括编写代码、测试和调试。

测试,在软件建成后,要对其进行测试,以确保其符合要求,功能正常。

部署,最后,软件被部署或发布给用户。

瀑布模型的优点如下：

(1)清晰的结构和明确的阶段:瀑布模型的各个阶段是明确的,每个阶段都有特定的可交付成果,在进入下一个阶段之前必须完成。这为软件开发过程提供了一个清晰的结构。

(2)易于理解：瀑布模型易于理解和遵循,使其成为软件开发项目的热门选择。

(3)文件驱动：瀑布模型非常强调文档,这对于保持项目需求、设计决策和其他重要信息的清晰记录非常有用。

瀑布模型的缺点：

(1)不灵活：瀑布模型是一种顺序性的方法,这意味着一旦一个阶段完成,就不能再重新审视。这可能会使它难以适应项目后期出现的变化或新的要求。

(2)有限的反馈:因为每个阶段都必须在进入下一个阶段之前完成,所以在项目的后期阶段,反馈和测试的机会有限。这可能导致问题在过程中被发现得太晚。

(3)耗费时间：瀑布模型可能很耗时,因为每个阶段都必须在下一个阶段开始之前完成。这可能使它难以对不断变化的需求或市场条件作出快速反应。

总的来说,瀑布模型对于有明确要求和清楚了解需要建立

的项目是很有用的。然而,对于那些需求可能会随着时间的推移而改变或演变的项目,它可能不是最好的选择。

2.2敏捷开发

敏捷开发是一种以客户需求进化为核心,采用迭代、循序渐进的方法进行软件开发。敏捷开发主张简单,拥抱变化,拥有可持续性,有递增变化。敏捷开发需要有目的的建模以及多种模型,敏捷开发需要快速反馈和随机应变的能力。

其优点如下：

(1)灵活性：敏捷开发被设计成具有灵活性和适应性,允许团队快速响应不断变化的需求或市场条件。

(2)持续的反馈和改进：敏捷开发强调持续的反馈和改进,在整个开发过程中定期审查和迭代。

(3)更快的上市时间:敏捷开发可以使软件产品更快地进入市场,因为功能可以在较小的、增量的版本中交付。

敏捷开发的劣势：

(1)缺少可预测性:敏捷开发的可预测性可能低于其他开发方法,因为项目的要求和范围可能随时间而改变。

(2)需要有经验的团队成员:敏捷开发需要有经验的团队成员,他们能够协同工作并适应不断变化的需求。

(3)范围蠕变的可能性：如果不仔细管理和控制需求,敏捷开发可能导致范围蠕变。

## 3 开发示例

3.1使用TDD开发

TDD的基本思路就是通过测试来推动整个开发的进行,但测试驱动开发并不只是单纯的测试工作,而是把需求分析,设计,质量控制量化的过程。

TDD的重要目的不仅仅是测试软件,测试工作保证代码质量仅仅是其中一部分,而且是在开发过程中帮助客户和程序员去除模棱两可的需求。TDD首先考虑使用需求(对象、功能、过程、接口等),主要是编写测试用例框架对功能的过程和接口进行设计,而测试框架可以持续进行验证,其主要步骤如下：

- 首先明确代码需要完成的功能
- 并新增一个测试
- 运行测试,若通过,则完成第六步
- 对功能代码进行改动
- 重新测试保证通过
- 优化代码并消除重复设计

2使用unittest模块测试：

几个概念：

TestCase 也就是测试用例

TestSuite 多个测试用例集合在一起

TestLoader是用来加载TestCase到TestSuite中的

TestRunner是来执行测试用例的,测试的结果会保存到TestResult实例中,包括运行了多少测试用例,成功了多少,失败了多少等信息。

```
def cal(a,b):
    return a+b#
    # res = cal(1,9)
    # if res == 3:
    #     print('成功')
    # else:
    #     raise Exception('测试失败')
    # unittest python中的单元测试框架import unittest
    class MyTest(unittest.TestCase):#继承TestCase
    def test_a(self):#函数的名字必须以test开头
    res = cal(1,2)
    self.assertEqual(3,res,msg='预期结果和实际结果不一致')
    def test_b(self):
    res = cal(9,3)
    self.assertEqual(3,res,msg='预期结果和实际结果不一致')
    unittest.main()  #运行测试用例
```

两种生成测试报告的方式：安装BeautifulReport模块,把HTMLTestRunner.py 放到pyCharm的External Libraries中。

```
import unittestimport HTMLTestRunnerimport BeautifulReport
    class MyTest(unittest.TestCase):
    def test_reg(self):
    '''注册'''#加这一行会在报告中显示用例描述
    print('reg')
    self.assertEqual(1,1)
    def test_log(self):
    '''登录'''
    print('log')
    self.assertEqual(1,1)
    def test_buy(self):
    self.assertEqual(1,2,msg='购买失败')
    def test_z(self):
    self.assertIn(1,[1,2,3])
    def test_assert(self):
    res = False
    self.assertFalse(res,)
    #下面这一坨是用HTMLTestRunner,产生不好看的测试报告
    f=open('report.html','wb')
    runner=HTMLTestRunner.HTMLTestRunner(f,title='lhtest',description='XX接口测试')#实例化一个runner,帮你跑测试用例
    suit = unittest.makeSuite(MyTest)#MyTest类变成测试集合runner.run(suit)
    #下面这一坨是用BeautifulReport,产生不好看的测试报告
```

```
suite=unittest.makeSuite(MyTest)#变成测试集合
report=BeautifulReport.BeautifulReport(suite)
report.report(filename='bfreport.html',description='接口测试报告')
# 测试集合：testsuite多个用例放在一起
# unittest  1、函数名必须是test开头,unittest才会帮你执行
# 2、用例运行的顺序是按照函数的首字母排序的,a-z
# testrunner运行测试用例的
# testloader查找测试用例的
```

如果我们有很多个模块,每个模块下面都写了很多python文件,每个python文件里面都有测试用例,那怎么把这个目录下的用例都执行了呢,就要先找到这个目录下的所有python文件,然后找到里面的测试用例,逐个执行,代码如下：

```
import unittest,HTMLTestRunner
suite=unittest.TestSuite()#创建测试套件
all_cases = unittest.defaultTestLoader.discover('.','test_*.py')
#找到某个目录下所有的以test开头的Python文件里面的测试用例
for case in all_cases:
suite.addTests(case)#把所有的测试用例添加进来
fp = open('res.html','wb')
runner=HTMLTestRunner.HTMLTestRunner(stream=fp,title='all_tests',description='所有测试情况')
runner.run(suite)
#运行测试
参数化
安装nose-parameterized模块
import unittestfrom parameterized import parameterized
def login(username,password):
print(username,password)
print('==============')
return 1
class MyTest(unittest.TestCase):
@parameterized.expand(
[
['xiaodong','123','success'],
['yangfan','456','fail'],
['hailong','1273','success'],
['liurongxin','1273','success'],
]
)
def test_login(self,username,passwd,check):
'正常登陆'
res=login(username,passwd)
self.assertEqual(check,res)
unittest.main()
```

setUp、tearDown、setUpClass、tearDownClass

①setup():每个用例执行前运行

②teardown():每个用例执行后运行

③setUpClass():必须使用@classmethod 装饰器,所有用例执行前运行一次

④tearDownClass():必须使用@classmethod装饰器,所有用例执行后运行一次

3.2使用doctest测试

```
python -m doctest-v test.py
```

但是在docstring中并非所有的内容都是测试用例,docstring中还包含其他文本,那怎么去区分这些内容呢？在docstring中测试用例以提示符>>>开始,以空行或者下一个>>>结束,介于中间的文本会被忽略。

修改后的内容测试结果是一样的。

处理不可预测的输出:有些情况下,可能无法预测准确的输出,但是依然可以进行测试。例如,获取某个对象的ID,每次运行测试的时候,得到的ID都是不一样的。

```
def identity(obj):
"""
>>> identity(1)
23400792
"""
return id(obj)
```

## 4 结语

Python是具有跨平台性质的计算机程序设计语言,而敏捷开发也是新一代软件开发的重要方法,基于Python进行敏捷开发能够节省时间和金钱与不必要的工作。

## [参考文献]

[1]贾勇.敏捷开发在信息管理系统设计中的应用研究[J].电脑知识与技术,2021,17(19):65-66+75.

[2]袁瑶佳.面向敏捷开发的需求管理系统的设计与实现[D].西安电子科技大学,2021.

[3]张洪阳.敏捷开发任务管理系统的设计与实现[D].华中科技大学,2019.

[4]韩亚平.敏捷开发管理平台的设计与实现[D].哈尔滨工业大学,2018.

[5]陆国浩,潘祺麟.基于敏捷开发的高职院校开放实验室系统设计[J].科技创新与生产力,2018,(03):118-120.