Auto Test设计文案

单元测试自动生成工具

曹磊

2020

目录

[一，背景 2](#_Toc34601001)

[1. 测试驱动开发（TDD） 2](#_Toc34601002)

[2. 自动化单元测试 2](#_Toc34601003)

[二，开发规划 3](#_Toc34601004)

[1. 设计目标 3](#_Toc34601005)

[2. 开发环境与工具 3](#_Toc34601006)

[3. 主要软件需求 4](#_Toc34601007)

# 一，背景

自动化测试现在软件工程中是普遍的，它可以保证一个项目得代码稳定性以及可检验性。便于新功能得开发，以及历史问题得修改。

自动化测试使得程序员无需在往一个庞大且存在了很长时间的C++系统中加入新的功能通常很耗时，而且无法估计进度。只是理解一段代码然后修改几行，有可能要花费几个小时甚至几天。开发者需要花上数小时等待代码改动编译完成，并且要等待更长的时间去检测这些改动是否和现有的系统完好集成。它可以更方便程序员确保程序在向目的地前进。

## 测试驱动开发（TDD）

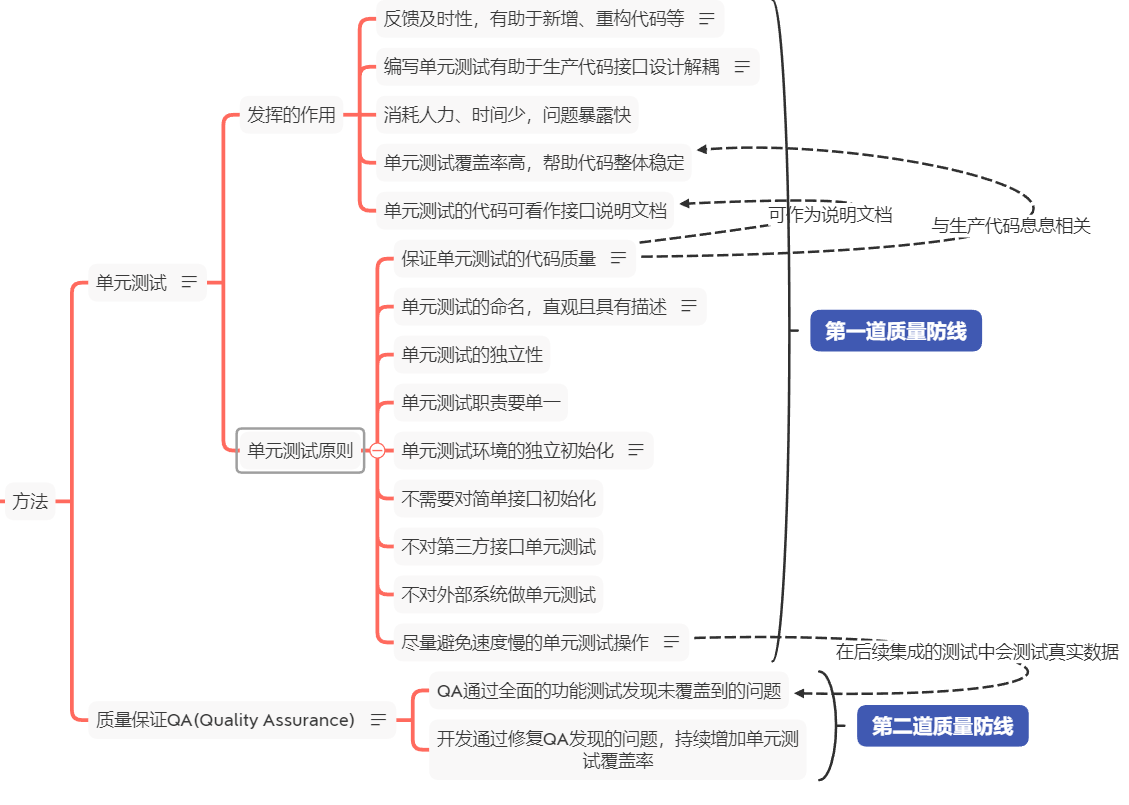
自动化测试分为几种模式，其中比较流行的模式是测试驱动开发，俗称TDD。它的方法是通过理解需求，先编写测试代码，然后再编写生产代码，努力去使通过的测试的开发模式。先编写测试代码在更深层次上促使编写的代码更加的遵循可测性。

TDD的缺点同样也很明确，它不是对付旧系统的良方。虽然可以在开发系统新特性时使用TDD，但通常依然需要清理系统中多年积累的障碍，尤其是这些障碍没有被自动测试代码覆盖到。为了进行这样的清理工作，除了使用TDD的技巧外，需要额外的战略和战术。

## 自动化单元测试

自动化单元测试的重点在“单元”上，单元指的是一个基本的功能其可以单独运行，目的明确。而大多数的需求都是由这种基础单元堆砌实现，因此保证每个单元的稳定与正确，是一个功能可以正常工作的基础。

下图为自动化单元测试的作用与原则：



# 二，开发规划

## 设计目标

本设计文档主要目的是设计一款可以依靠可视化的流程图、代码结构来进行自动化单元测试的代码生成以及执行。与当前市面上通用的Catch2、GTest、CPPUnitTest等，自动化测试工具最大的区别就是支持界面配置，同时可以可视化的生成Mock函数、组件等等。

## 开发环境与工具

1. **开发环境**

（1），开发系统使用Linux操作系统(Ubuntu 18.04)，对于搭建项目更加透明，没有那么多IDE做了太多隐藏的事情。

（2），界面开发使用QT5.13.2 版本库，相较于5.14 版本更加稳定，特性相差也并不大，支持C++11标准。

（3），项目管理主要通过CMake搭建而不使用QMake，主要优势是CMake更加的通用同时搭建非QT模块以及项目时更加通用。

（4），Linux下GCC 7.5.0版本编译器，Cmake版本3.10.2。Windows下MSVC 19.16版本编译器，Cmake版本3.12.18。

1. **编辑工具**

编辑工具未使用QT官方IDE或Visual Studio，原因同使用Cmake相同，期望环境搭建尽量通用。推荐使用VS Code编辑器，可以通过插件增加想要的功能，同时它没有做多余的事情，搭建环境可以让你对所处的编译环境了如指掌。

## 主要软件需求

1. **界面需求**



整体界面布局分为四部分：

（1），上方控制栏，包含基础的控制、设置、信息等功能按钮。

（2），流程图预览区主要展示需要测试的代码块调用逻辑，包括详细的接口调用逻辑，可以使用mock进行打桩或者调用实例进行测试。例如：实现一个登陆功能的类，需要调用网络接口、界面读取接口（获取账号、密码）、组包接口等等，那么流程图中可以从大类上将这些接口的调用逻辑进行描绘，双击进入某个具体接口的实现又包括实现的调用逻辑，可以便于细分单元测试的接口。

（3），编辑区，当对某个接口具体进行测试时，使用预先生成的测试代码块，当代码块无法满足需要，可以使用编辑区修改测试的具体内容。

（4），功能区，当前功能区主要mock桩函数自动实现。可以预先包含几种常见的桩函数模板类型，也可以通过mock生成工具来自定义想要生成的模板类型。生成后将会自动显示在编辑区，也可以通过编辑区来进行修改来满足测试需求。