1. 需求和目标

我觉得，其实自动化测试跟其它任何一种测试类型（比如异常测试、稳定性测试、性能测试等）都是类似的，它也是一种测试类型而已。在开展测试之前，我们首先必须要明确自动化测试的需求是什么，要解决什么样的问题。

1.1 让“自动化”代替“手动”

在我看来，初期的自动化测试，我的目标很明确，我就是要让“自动化”代替“手动”，让自动化真正地跑起来，凡是“自动化”跑过的内容，我绝不再去手工重复执行一遍。这样至少我有一个很明确的收益：每完成一条自动化用例，我减少了一条手工用例的执行时间。

【必须要提醒的是，让“自动化”完全替代“手动”，其实对自动化用例的稳定性、容错都有一定的要求。你要花一定时间去思考用例执行过程中的异常场景，是否足以充分替代手工测试。因此，我在增加用例的时候都会非常谨慎，确保用例集是稳定100%通过的前提下才会增加新的用例。】

对于正常情况下（排除环境、开发代码的问题）有时100%通过，有时90%通过的自动化用例集，我觉得它的作用和参考价值为0。正常的用例集就应该是100%通过的。

1.2 让“回归”自动化

上节说了让“自动化”替代“手动”，每完成一条自动化用例都是有明显收益的。那如何让收益最大化呢，当然是让每次回归或上线验证“不得不”执行的用例优先自动化。如果完成了回归用例集的全部自动化，那我就可以用它来替代我的日常回归，和上线回归工作，极大地释放我的手工验证时间。

这里必须要指出的是，我跟的项目其实是一个对系统稳定性的要求要高于新功能的引入的一个后台项目，所以它的核心功能是比较固定的，其实大多数后台项目也是类似的，核心功能聚合、对系统的稳定性要求高。这就需要保障系统的核心功能完善。所以我们可以先将“核心功能”的验证完成自动化。

1.3 不要让环境成为瓶颈

前面说了，旧的用例集在维护的过程中给测试人员增加了很多额外的负担，到最后发现很多都是环境的问题。当时的情形就是专门搭建了另一套测试环境专门用于自动化测试，而大数据的后台环境搭建和维护非常的复杂，如果同时维护多套环境，难免会在一些组件升级的过程中出现遗漏，导致环境不同步。因此，我们的自动化测试用例前期完全可以直接在功能测试环境执行，因为功能测试环境肯定是会一直随着版本的迭代向前不断更新的。

2. 技术选型

在明确了目标后，要开始技术选型。常见的自动化测试类型，包括

- 接口自动化
- UI自动化
- 基于shell交互命令执行的自动化

此外，不属于测试范畴，但是也可以实现自动化、释放手工时间的还有

- 数据准备自动化
- 环境编译、部署、打包自动化
- 稳定性测试/性能测试结果指标获取、校验自动化
- 机器资源监控、报警自动化
- 其它所有手工重复执行的操作

这里再插一段题外话：有些人可能会疑惑，现在其实有很多接口测试平台，测试人员可以直接在平台上完成接口测试，在选型时怎么抉择？——这里我不评价哪种方式更好，只想说下自己的看法：我觉得两种其实各有各的好处：

- 编写代码的方式：

优点：提升自己的编码能力，问题定位能力，具备更高的灵活性和可操作性。 缺点：结果展示不直观，不易于协作。其他人维护代码困难，难以推动开发执行。

- 接口平台的方式：

优点：简便，上手容易，可以在项目组间很好的协作和维护，测试记录和结果一目了然。 缺点：离开了平台，可能又要回归手动。

对于测试人员而言，如果有精力和时间的话，我建议是两种都要掌握，甚至是自己去开发接口测试平台的能力。

3. 自动化实施过程

目前我跟的项目里已经实现自动化的内容包括：基于接口的场景回归自动化测试、编译部署过程自动化、Jacoco覆盖率统计并接入CR平台（代码变更分析平台）的自动化、对外/上线打包发布的自动化、稳定性测试结果校验的自动化。

下面着重介绍下项目的接口自动化框架的搭建和设计过程。

3.1 准备工作

老生常谈，开始自动化前，我仍然想再次强调一定要明确自己的需求是什么。在我的项目里，我的需求主要有以下几点：

- 同一份代码可以在多个集群执行

- 各个集群的测试数据相互独立，不会互相影响
- 可以方便地与数据库进行交互
- 当用例执行出错时，有详细的日志帮助定位
- 较好的可维护性和集群扩展性。