从事软件测试行业，每天面对的被测对象都是软件。如果想要更好的去完成测试工作，首先需要对被测对象，也就是对软件要有基本的了解。

软件

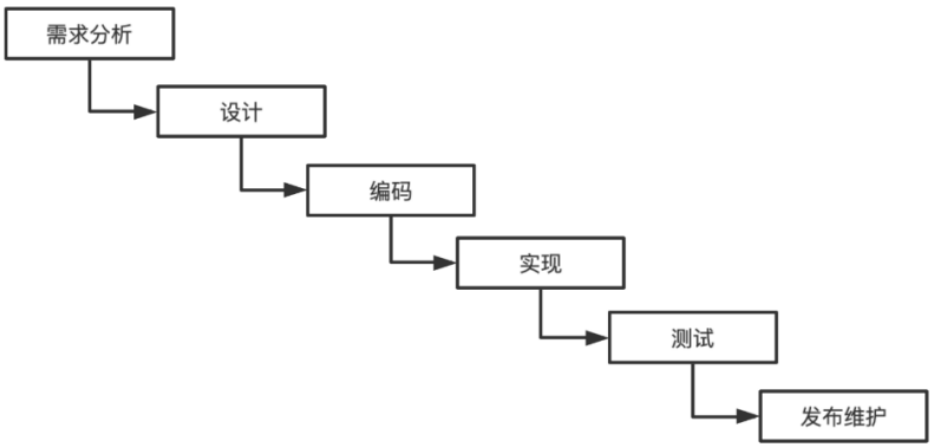
与计算机系统操作有关的计算机程序、可能有的文件、文档及数据。
程序好理解，就是可以操作的产品。比如 wps、微信、QQ、网页等等这些都是程序。比如说需求文档、设计文档、用户手册这些东西都属于文档。在页面中展示的，还有用户输入的内容这些都是数据。
所以说程序、文档、数据这三个结合起来，就是完整的软件。

软件开发流程的演变

流程的演变其实就是软件开发模型的演变过程。
软件开发模型就是在软件开发当中，逐渐总结了很多的经验，这些经验经过提炼总结就变成了开发模型。比如最开始的瀑布模型，后来到了敏捷开发模型，一直发展到现在最火的 DevOps 模型。
下面，分别介绍一下这几种开发模型。

传统瀑布模型

瀑布大家都熟悉，水是从上到下的流下来的。那瀑布模型也是一样，像水流一样从上往下一步一步进行的。



需求分析

不管做任何事情，分析的工作是肯定是必不可少的。瀑布模型里面也是这样，首先要做的就是需求分析。
需求文档是产品人员从用户那里了解并搜集到的。了解清楚用户想要什么之后，再把它细化成为一个文档。文档会清楚列出系统大致的大功能模块，大功能模块有哪些小功能模块，并且还列出相关的界面和界面功能。有了这个文档，产品的 UI 界面、功能就都确定下来了。

设计

需求分析之后就开始做设计，需要设计的包括两个方面：

- 界面设计：UI 设计师根据需求设计出来前端界面的一个设计稿
- 程序设计：设计基本业务处理流程，模块怎么划分，接口的规范等等

都设计好了之后，开发人员就可以进入编码的阶段了。

编码
在软件编码阶段，开发会根据设计好的方案，把这些方案通过代码去进行实现。

实现
实现就相当于开发的代码已经实现了需求里面的这些功能了。

测试
实现之后测试人员就可以介入了。这就是瀑布模型的流程，有了代码，再去做测试。

发布维护
测试工作完成之后，再发布上线，并且继续维护产品。

特点
在瀑布模型中，软件开发的各项活动严格按照线性方式进行，当前活动接受上一项活动的工作结果，当前活动的工作结果需要进行验证。

瀑布模型是线性模型的一种。它在所有的开发模型当中占有重要的地位，是所有其他模型的一个基础。其他的模型都是根据这个线性模型演变过来的。

瀑布模型的优点很明显，开发的各个阶段比较清晰，强调早期计划及需求调查，比较适合需求稳定的产品开发。

但是因为开发模型是线性的，增加了开发的风险，所以早期的错误可能要等到开发后期的阶段才能发现。

为了解决瀑布模型里面的这些问题，后面又慢慢发展出来了别的开发模型。

敏捷开发模型

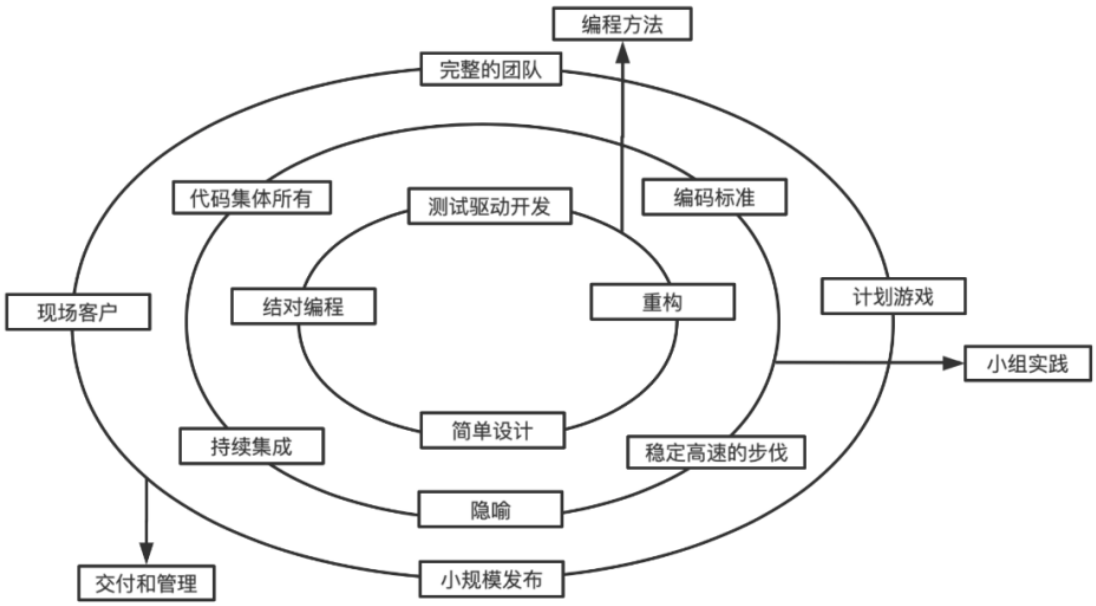
敏捷开发模式是一种从 90 年代开始逐渐引起广泛关注的一些新型软件开发方法。这种开发模型更适用于需求频繁变化和需要快速开发的场景。

常见的敏捷开发模型有 XP 和 Scrum，下面分别介绍下这两种开发模型。

XP 极限编程

XP（eXtreme Programming）是一种近螺旋式的开发方法。它是把复杂的开发过程分解为一个个相对比较简单的小周期。在每一个周期里面，项目人员和客户都可以非常清楚开发进度、变化、待解决的问题和潜在的困难等，而且可以

根据实际情况及时地调整开发过程。



在上图中可以看出，极限编程是从 3 个维度去组织开发流程的。

编程方法

首先是编程方法这个维度。在这个纬度当中，对开发人员的开发方法做出了规定。

- **简单设计**：XP 要求用最简单的办法实现每个小需求。这些设计只要能满足客户在当下的需求就可以了，不需做更高深的设计，这些设计都将在后续的开发过程中可以不断地调整和优化。
- **结对编程**：指代码由两个人一起完成。一个人主要考虑编码细节。另外一个人主要关注整体结构，不断的对第一个开发写的代码进行评审。
- **测试驱动开发**：测试驱动开发的基本思想就是在开发功能代码之前，先编写测试代码。测试代码编写好了之后，再去编写可以通过测试代码的功能代码。这样就可以让测试来驱动整个开发过程的进行。这样做，有助于编写简洁可用和高质量的代码，有很高的灵活性和健壮性。
- **重构**：XP 强调简单的设计，但简单的设计并代表是没有任何结构的流水，也不是缺乏重用性的程序设计。XP 提倡重构代码，主要是努力减少程序和设计中重复出现的部分，增强程序和设计的可重用性。

小组实践

小组实践是从团队合作的维度去规定工作方法。

- **代码集体所有**：代码集体所有意味着每个人都对所有的代码负责。反过来又意味着每个人都可以更改代码的任意部分。
- **编码标准**：因为大家可以都可以改代码，那开发小组中的所有人都需要遵循一个统一的编程标准。这样所有的代码看起来好像是一个人写的。因为有了统一的编程规范，每个程序员更加容易读懂其他人写的代码，这是实现代码集体所有的重要前提之一。
- **稳定高速的步伐**：团队只有持久才有获胜的希望。可以把项目看作是马拉松长跑，而不是全速短跑。需要团队成员保持长期稳定的工作节奏。
- **持续集成**：集成就是要把大家的代码合并到一起。团队开发成员需要经常集成它们的工作。每次集成都通过自动

化的构建（这其中还包括了自动化测试)来验证，这样才能尽快地发现集成错误。

- **隐喻**：为了帮助每个人一致清楚地理解要完成的客户需求、要开发的系统功能，团队需要用很多形象的比喻来描述系统或功能模块是怎样工作的。比如，对于一个搜索引擎，它的系统隐喻可能就是“一大群蜘蛛，在网上四处寻找要捕捉的东西，然后把东西带回家中。”

交付和管理

最后一个就是发布管理的维度了。交付是把产品交到客户手上。发布就是把产品上线，让用户可以访问。总体来说，交付和发布都是让用户可以拿到产品去使用。

- **小规模发布**：那规模有多小呢？就是每个迭代 1-3 周时间。在每个迭代结束的时候，团队交付可运行的，经过测试的功能，这些功能可以马上投入使用。
- **计划游戏**：预测在交付日期前可以完成多少工作，确定现在和下一步该做些什么。不断的回答这两个问题，就是直接服务于如何实施及调整开发过程。
- **完整的团队**：每一个项目贡献者都是“团队”完整的一部分。这个队伍是围绕着一个每天和队伍坐在一起共同工作的商业代表——“客户”建立起来的。
- **现场客户**：在 XP 中，“客户”并不是为系统付账的人，而是真正使用该系统的人。XP 认为客户应该时刻在现场解决问题。

从 XP 开发模型可以看出来，里面开发和客户是占据主导地位的。测试的工作基本都是通过自动化的方式来进行。比如在编码过程中的测试驱动开发这个环节，还有持续集成中也包含了自动化的测试。总体而言这个开发模型对开发和测试的要求都是非常高的，团队里面的人必须都有非常高的水平，这个模型才能运转成功。这是开发小型项目的一个理想状态下的情况，比较难实现。

SCRUM

在 Scrum 模型里面，最基本的概念是 Sprint。Sprint 其实就是一个冲刺，通俗一点来说就是一个迭代周期。

整个项目开始之前，会先有一个产品 Backlog。使用产品 Backlog 来管理产品的需求的。它是整个项目的概要文档。Backlog 是一个按照商业价值排序的需求列表，列表条目的体现形式通常为用户故事。

Scrum 团队从产品 Backlog 中挑选最高优先级的需求进行开发。挑选的需求在 Sprint 计划会议讨论。

在 Sprint 上经过讨论、分析和估算得到相应的任务列表，可以称为 Sprint Backlog。

Scrum 中，整个开发过程由若干个短的迭代周期组成，一个短的迭代周期称为一个 Sprint，每个 Sprint 的建议长度是二至四周。

在每个迭代周期中，Scrum 团队会举行每日站会。在每日站会上检验 Sprint 目标的进展，做出调整，从而优化次日的工作。

每个迭代结束时，Scrum 团队将递交潜在可交付的产品增量。

在每个迭代周期最后，需要进行一次 Sprint 评审会议，让团队向产品负责人和利益相关者展示已完成的功能。

Sprint 评审会议结束之后，下一个 Sprint 计划会议之前，需要进行 Sprint 回顾会议。回顾会议是要找出 Sprint 过程中，哪些地方执行的很好，哪些地方执行的不好，团队可以做哪些改进。
这就整个 SCRUM 模型的工作流程。在每一个 Sprint，也就是一个迭代周期中，其实是一个小的瀑布。在每个迭代周期中，都会完成一个从需求分析 - 设计 - 编码 - 测试 - 上线这样的完整流程。不同的迭代周期可能是部分重合的。比如说第一个迭代周期进行到了测试阶段，第二个迭代周期的需求分析可能已经开始了。这样不停的循环迭代往下进行。

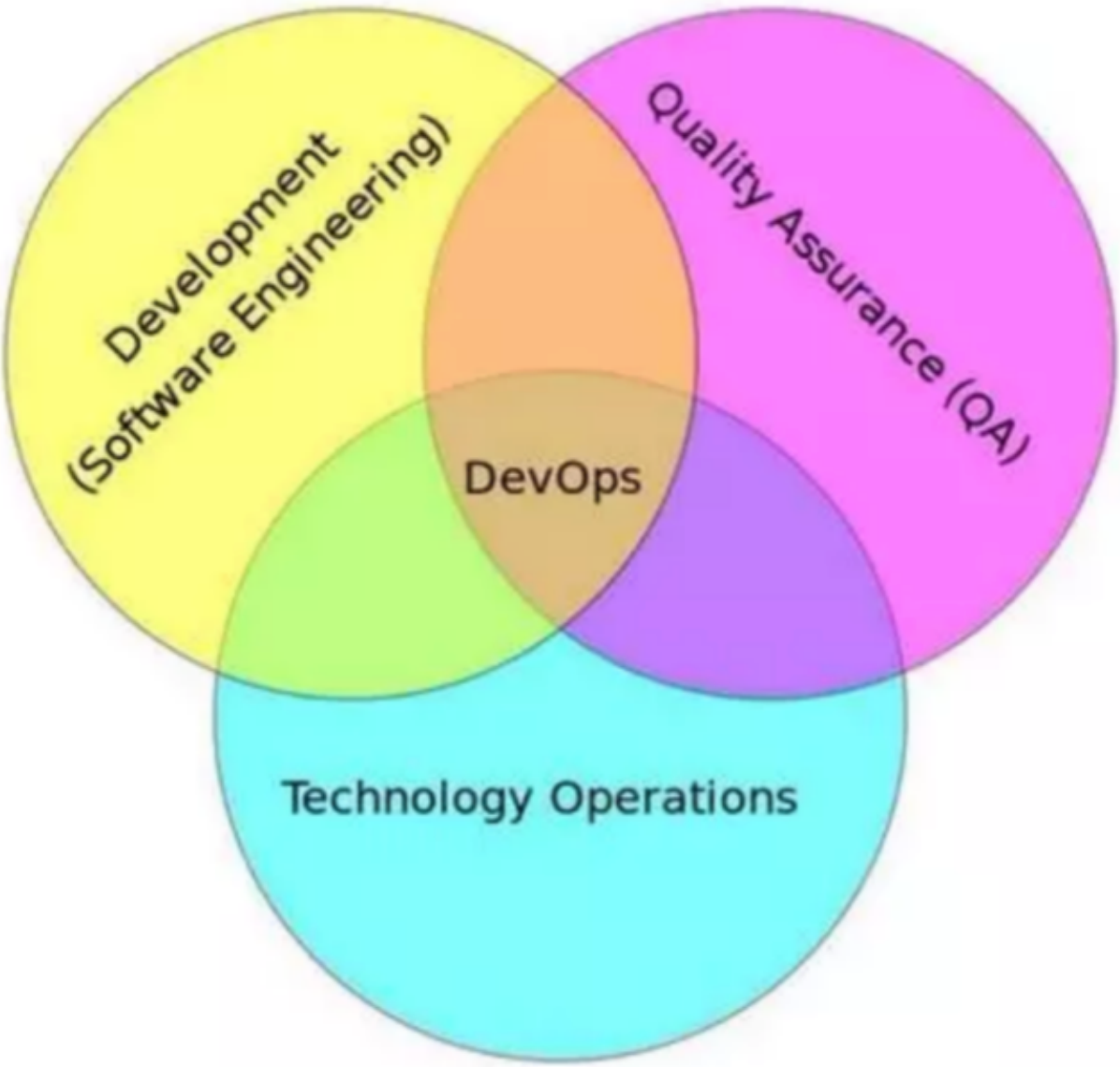
DevOps开发模型

DevOps（Development 和 Operations 的组合词），它涉及软件在整个开发生命周期中各个阶段。

DevOps 是非常关注开发（Dev）、运维（Ops）、以及测试人员之间沟通合作的一个开发模型。

在 DevOps 里，是通过自动化的软件交付的流程，来让构建、测试、发布软件能够更加地快捷、频繁和可靠。

它的出现其实就是因为现在的软件需要更加快速的上线，如果想实现每天都能上线新功能。但是敏捷开发模型，它再快也得一周的时间，实现不了这个需求。所以大家意识到了，为了能够更加快捷的上线，开发、测试和运维工作必须紧密合作。所以说 DevOps 更适合使用在需求频繁变化、开发、测试运维都需要敏捷的场景下。



DevOps生命周期

下面来看看 DevOps 模型中又包含了哪些阶段。

持续开发

这是 DevOps 生命周期中软件不断开发的阶段。与瀑布模型不同的是，软件可交付成果被分解为短开发周期的多个任务节点，在很短的时间内开发并交付。

这个阶段包括计划、编码和构建阶段

- 计划阶段：可以使用一些项目管理工具，比如 JIRA 来管理整个项目
- 编码阶段：可以使用 Git 或者 SVN 来维护不同版本的代码
- 构建阶段：使用打包工具，比如 Maven 工具，把代码打包到可执行文件中

持续测试

在这个阶段，开发出来的软件会被持续地进行测试。

对于持续测试，可以使用一些自动化测试工具，比如说 Selenium、Appium。Selenium 是做 web 自动化的工具，Appium 是做 app 自动化的工具。自动化的工具还需要配合测试框架一起去使用，比如 Java 中的 TestNG、JUnit，

python 中的 unittest、pytest。有了这些自动化测试的工具，就可以持续的对开发出来的软件进行测试了。

在这个阶段，使用 Docker 容器实时模拟“测试环境”也是非常方便的。

持续集成（CI）

一旦新提交进来的代码测试通过，就会不断地与现有代码进行集成。这就是持续集成的过程了。

这个时候可以使用 Jenkins，这是现在最流行的持续集成的工具。使用 Jenkins，可以从 Git 库提取最新的代码，并生成一个构建，最终可以部署到测试或生产服务器。

还可以把 Jenkins 设置成发现 Git 库里有新提交的代码，就可以自动触发新构建，也可以在单击按钮时手动触发一个新的构建。有了 Jenkins 这款利器，就可以非常方便的完成持续集成的工作。

持续部署

持续集成完成之后，就可以直接把代码部署到各种环境中。在这个阶段，需要保证只有通过了持续测试的正确代码，才能被部署到服务器上。

因为如果上线了新功能，产品就会有更多用户去使用。这样的话，运维人员可能还需要扩展服务器来容纳更多用户。如果可以实现持续部署，就可以通过配置管理工具快速、频繁地执行部署任务。让产品能够更快的和用户见面。这就打通了开发、测试到上线的一个快速通道。

在这个阶段，容器化工具 Docker 也发挥着重要作用。它可以帮助保持各种环境是一致的。比如说测试环境、生产环境等等这些，因为环境的不同也可能会导致一些 Bug 出现。

持续监控

部署上线之后，就到了持续监控的阶段。这是 DevOps 生命周期中非常关键的阶段。通过线上的监控可以帮助提高软件的质量，监控软件的性能。

这里也会涉及运营团队的参与，他们也会监控用户在使用产品过程中的一些错误行为，为以后需求的进一步优化提供数据支持。

在这个阶段，可以使用 ELK Stack。这是一个搜集线上数据，并且分析展示的平台。通过这个工具可以自动的去搜集用户的动作，产品的一些线上的 bad case，通过分析这些数据，可以为产品将来的发展方向做出指导。

上面这些内容就是 DevOps 整个的生命周期。